SageMaker Studio Administration Best Practices

Publication date: **April 25, 2023** (*Document revisions*)

Abstract

Amazon SageMaker Studio provides a single, web-based visual interface where you can perform all machine learning (ML) development steps, which improves data science team productivity. SageMaker Studio gives you complete access, control, and visibility into each step required to build, train, and evaluate models.

In this whitepaper, we discuss best practices for subjects including operating model, domain management, identity management, permissions management, network management, logging, monitoring, and customization. The best practices discussed here are intended for enterprise SageMaker Studio deployment, including multi-tenant deployments. This document is intended for ML platform administrators, ML engineers, and ML architects.

Are you Well-Architected?

The AWS Well-Architected Framework helps you understand the pros and cons of the decisions you make when building systems in the cloud. The six pillars of the Framework allow you to learn architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems. Using the AWS Well-Architected Tool, available at no charge in the AWS Management Console, you can review your workloads against these best practices by answering a set of questions for each pillar.

In the Machine Learning Lens, we focus on how to design, deploy, and architect your machine learning workloads in the AWS Cloud. This lens adds to the best practices described in the Well-Architected Framework.

Introduction

When you administrate SageMaker Studio as your ML platform, you need best practices guidance for making informed decisions to help you scale your ML platform as your workloads grow. For provisioning, operationalizing, and scaling your ML platform, consider the following:

- Choose the right operating model and organize your ML environments to meet your business objectives.
- Choose how to set up SageMaker Studio domain authentication for user identities, and consider the domain-level limitations.
- Decide how to federate your users' identity and authorization to the ML platform for fine-grained access controls and auditing.
- Consider setting up permissions and guardrails for various roles of your ML personas.
- Plan your virtual private cloud (VPC) network topology, considering your ML workload's sensitivity, number of users, instance types, apps, and jobs launched.
- Classify and protect your data at rest and in transit with encryption.
- Consider how to log and monitor various application programming interfaces (APIs) and user activities for compliance.
- Customize the SageMaker Studio notebook experience with your own images and lifecycle configuration scripts.

Operating model

An *operating model* is a framework that brings people, processes, and technologies together to help an organization deliver business value in a scalable, consistent, efficient manner. The ML operating model provides a standard product development process for teams across the organization. There are three models for implementing the operating model, depending on the size, complexity, and business drivers:

- **Centralized data science team** — In this model, all data science activities are centralized within a single team or organization. This is similar to the Center of Excellence (COE) model, where all business units go to this team for data science projects.
- **Decentralized data science teams** — In this model, data science activities are distributed across different business functions or divisions, or based on different product lines.
- **Federated data science teams** — In this model, shared services functions such as code repositories, continuous integration and continuous delivery (CI/CD) pipelines, and so on are managed by the centralized team, and each business unit or product level function is managed by decentralized teams. This is similar to the hub and spoke model, where each business unit has their own data science teams; however, these business unit teams coordinate their activities with the centralized team.

Before deciding to launch your first studio domain for production use cases, consider your operating model and AWS best practices for organizing your environment. For more information, refer to Organizing Your AWS Environment Using Multiple Accounts.

The next section provides guidance on organizing your account structure for each of the operating models.

Recommended account structure

In this section, we briefly introduce an operating model account structure that you can start with and modify according to your organization’s operating requirements. Regardless of the operating model you choose, we recommend implementing the following common best practices:

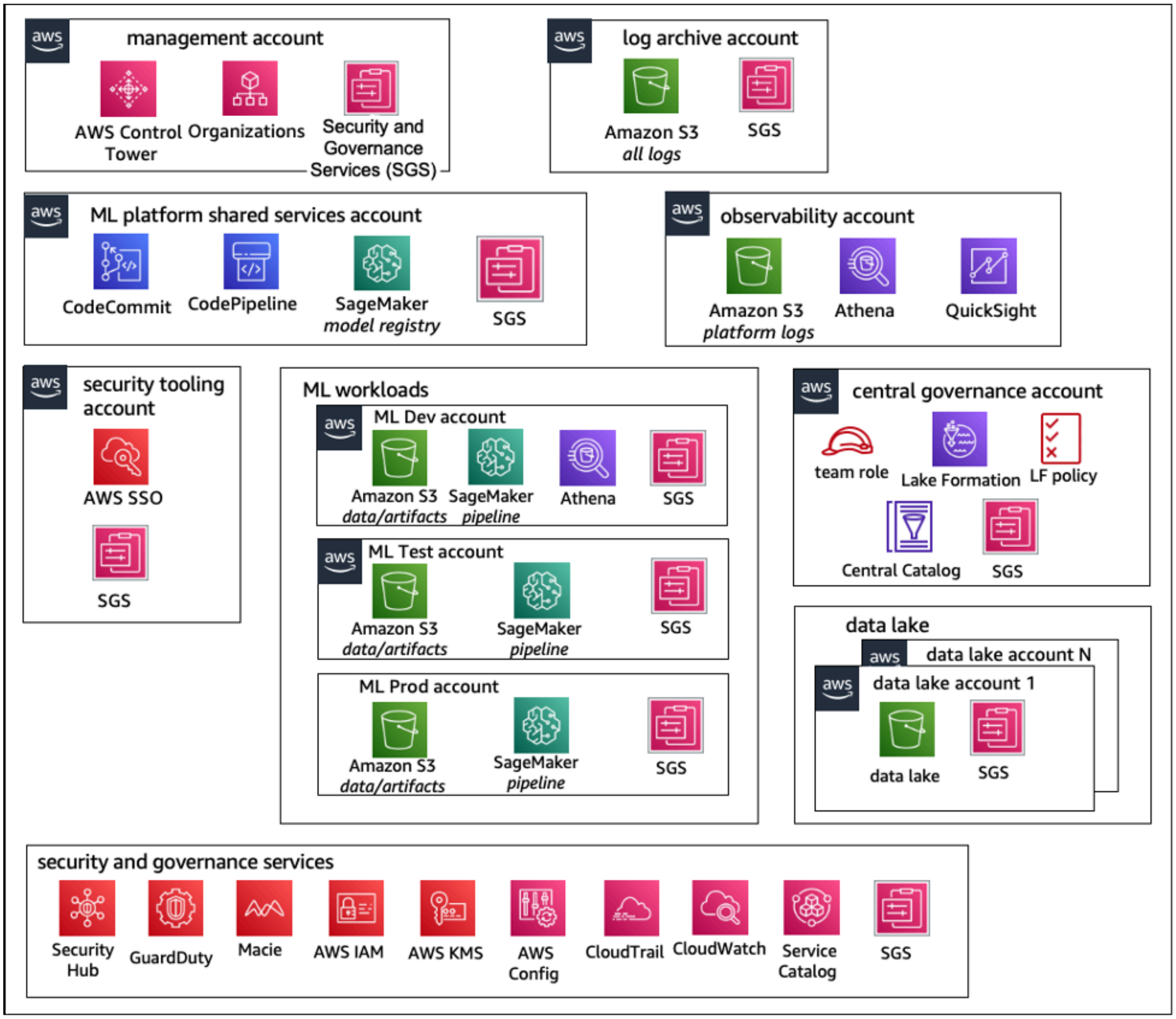
- Use AWS Control Tower for setup, management, and governance of your accounts.
- Centralize your identities with your Identity Provider (IdP), and AWS IAM Identity Center with a delegated administrator Securitiy Tooling account and enable secure access to workloads.

- Run ML workloads with account level isolation across development, test, and production workloads.
- Stream ML workload logs to a log archive account, and then filter and apply log analysis in an observability account.
- Run a centralized governance account for provisioning, controlling, and auditing data access.
- Embed security and governance services (SGS) with appropriate preventive and detective guardrails into each account to ensure security and compliance, as per your organization and workload requirements.

Centralized model account structure

In this model, the ML platform team is responsible for providing:

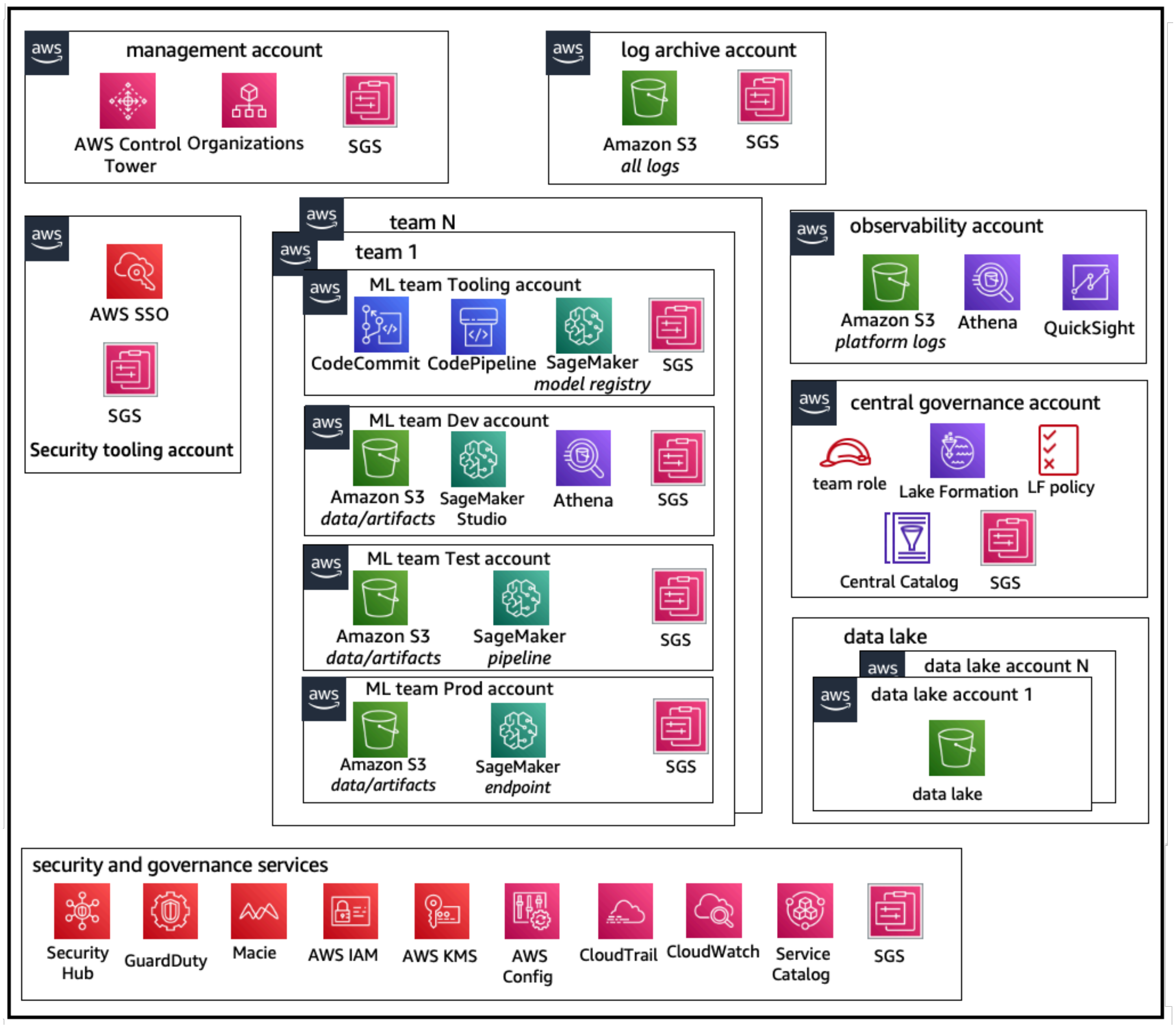
- A shared services tooling account that addresses the Machine Learning Operations (MLOps) requirements across data science teams.
- ML workload development, test, and production accounts that are shared across data science teams.
- Governance policies to ensure each data science team workload runs in isolation.
- Common best practices.



Centralized operating model account structure

Decentralized model account structure

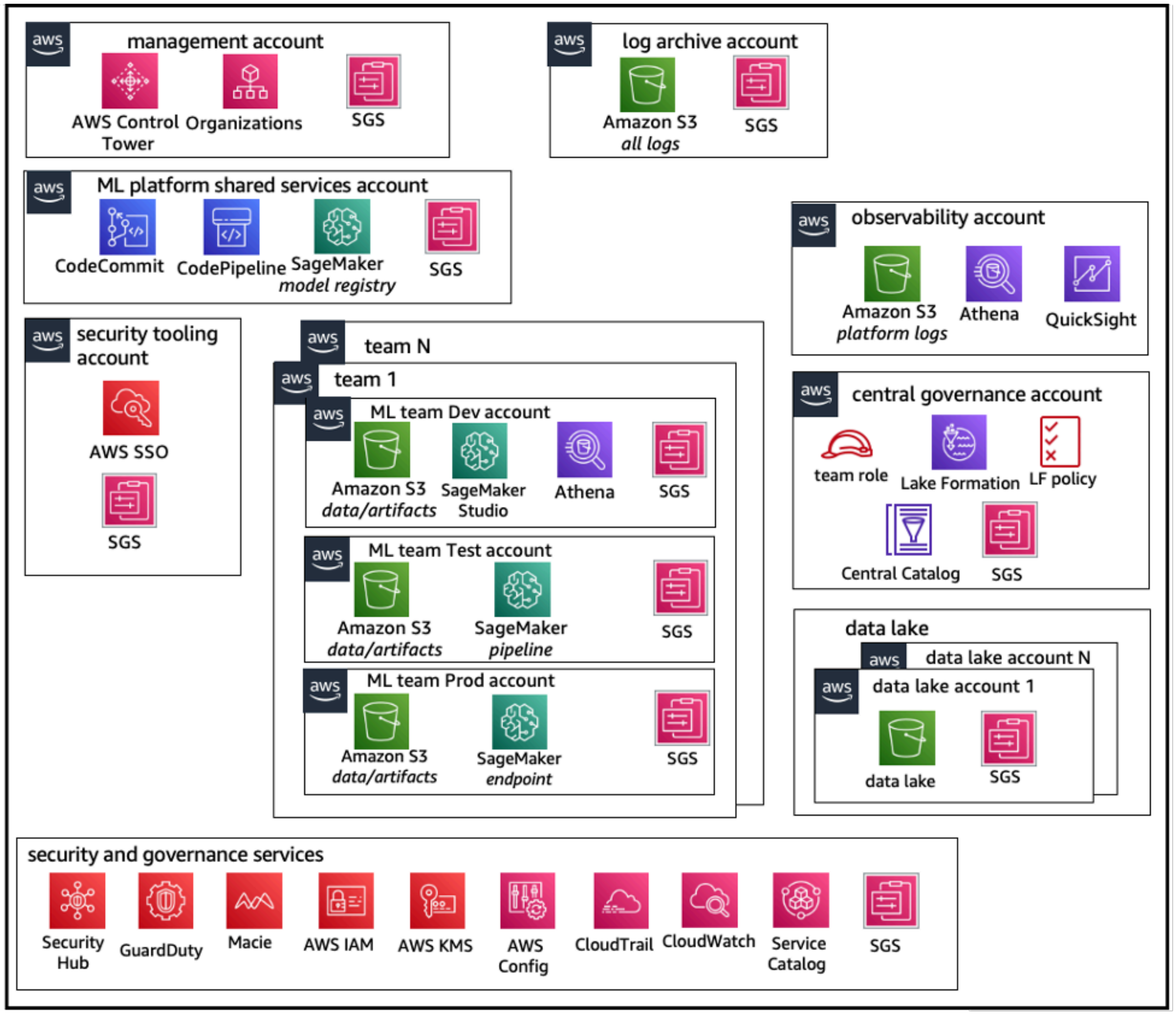
In this model, each ML team operates independently for provisioning, managing, and governing ML accounts and resources. However, we recommend ML teams use a centralized observability and data governance model approach to simplify data governance and audit management.



Decentralized operating model account structure

Federated model account structure

This model is similar to the centralized model; however, the key difference is that each data science/ML team gets their own set of development/test/production workload accounts that enable robust physical isolation of their ML resources, and also enable each team to scale independently without impacting other teams.



Federated operating model account structure

ML platform multitenancy

Multitenancy is a software architecture where a single software instance can serve multiple, distinct, user groups. A *tenant* is a group of users who share common access with specific privileges to the software instance. For example, if you are building several ML products, then each product team with similar access requirements can be considered a tenant or a team.

While it possible to implement multiple teams within a SageMaker Studio instance (such as SageMaker Domain), weigh those advantages against trade-offs such as blast radius, cost attribution, and account level limits when you bring multiple teams into a single SageMaker Studio domain. Learn more about those trade-offs and best practices in the following sections.

If you need absolute resource isolation, consider implementing SageMaker Studio domains for each tenant in different account. Depending on your isolation requirements, you may implement multiple lines of businesses (LOBs) as multiple domains within a single account and Region. Use shared spaces for near real-time collaboration between members of the same team/LOB. With multiple domains, you will still use identity access management (IAM) policies and permissions to ensure resource isolation.

SageMaker resources created from a domain are auto-tagged with the domain Amazon Resource Name (ARN) and the user profile or space ARN for easy resource isolation. For sample policies, refer to Domain resource isolation documentation. There you can see the detailed reference for when to use a multi-account or a multi-domain strategy, along with the feature comparisons in the documentation, and you can view sample scripts to backfill tags for existing domains on the GitHub repository.

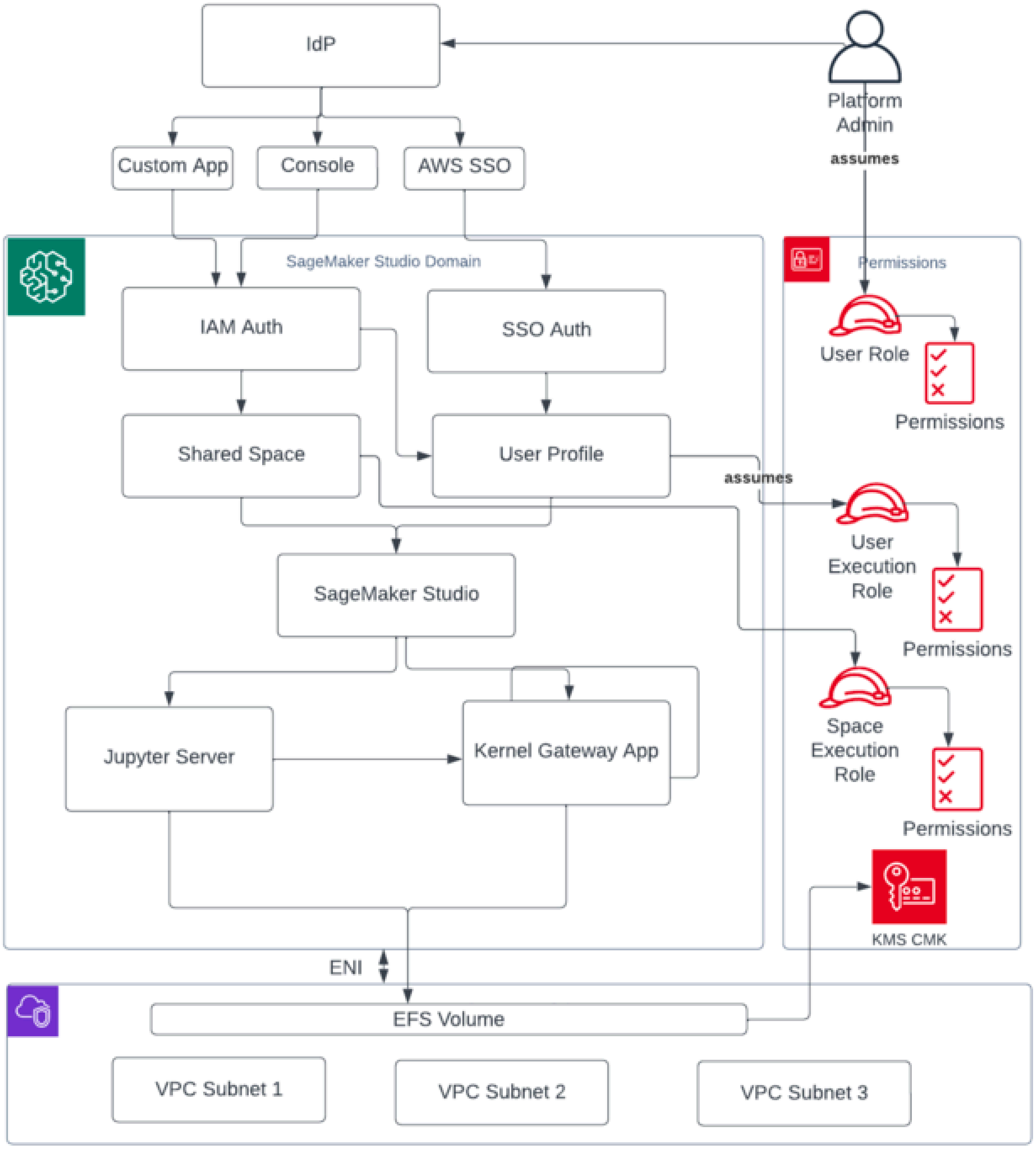
Finally, you can implement a self-service deployment of SageMaker Studio resources into multiple accounts using AWS Service Catalog. For more information, refer to Manage AWS Service Catalog products in multiple AWS accounts and AWS Regions.

Domain management

An Amazon SageMaker Domain consists of:

- An associated Amazon Elastic File System (Amazon EFS) volume
- A list of authorized users
- A variety of security, application, policy, and Amazon Virtual Private Cloud (Amazon VPC) configurations

The following diagram provides a high-level view of various components that constitute a SageMakerStudio domain:

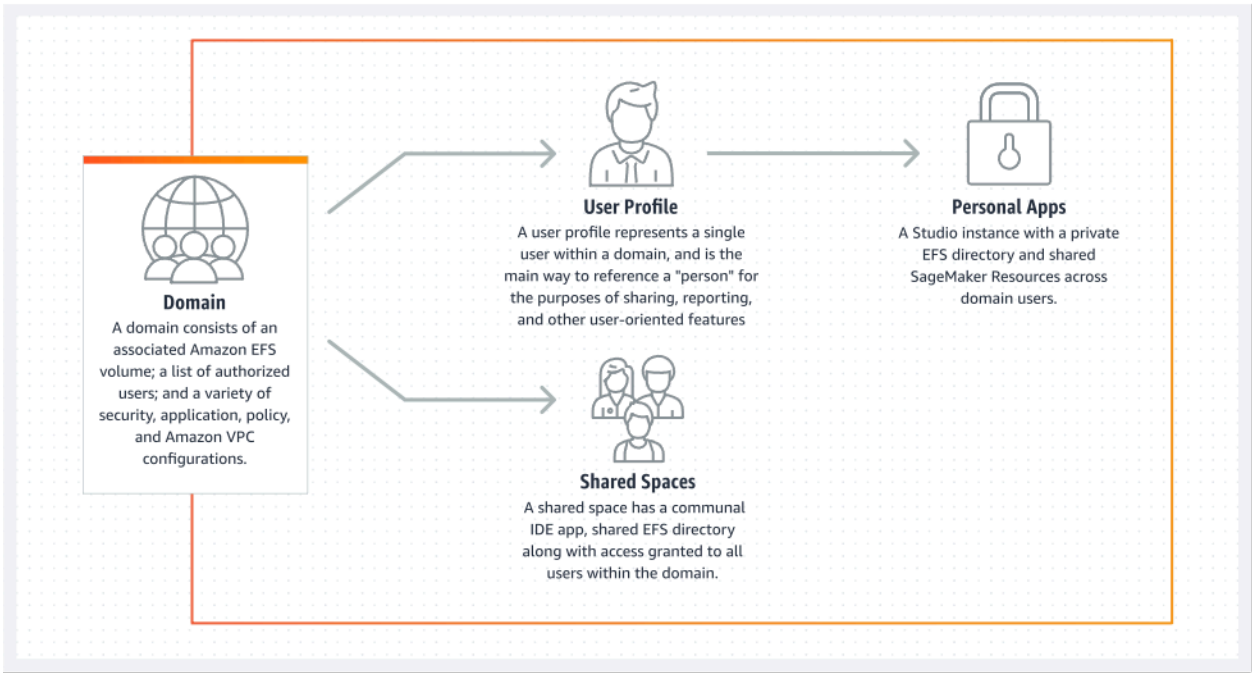


High-level view of various components that constitute a SageMaker Studio domain

Multiple domains and shared spaces

Amazon SageMaker now supports the creation of multiple SageMaker domains in a single AWS Region for each account. Each domain can have its own domain settings, such as authentication mode, and networking settings, such as VPC and subnets. A user profile cannot be shared across domains. If a human user is part of multiple teams separated by domains, create a user profile for the user in each domain. Refer to the Multiple Domains Overview to learn about backfilling tags for existing domains.

Each domain set up in IAM authentication mode can make use of shared space for near real-time collaboration between users. With a shared space, users get access to a shared Amazon EFS directory, and a shared JupyterServer app for the user interface, and can co-edit in near real-time. Automatic tagging of resources created by shared spaces allows the administrators to track costs on a project level. The shared JupyterServer UI also filters resources such as experiments and model registry entries so that only items relevant to the shared ML endeavor will be shown. The following diagram provides an overview of private apps and shared spaces within each domain.


Domain
A domain consists of an associated Amazon EFS volume; a list of authorized users; and a variety of security, application, policy, and Amazon VPC configurations.
User Profile
A user profile represents a single user within a domain, and is the main way to reference a "person" for the purposes of sharing, reporting, and other user-oriented features
Personal Apps
A Studio instance with a private EFS directory and shared SageMaker Resources across domain users.
Shared Spaces
A shared space has a communal IDE app, shared EFS directory along with access granted to all users within the domain.

Overview of private apps and shared spaces within a single domain

Set up shared spaces in your domain

Shared spaces are typically created for a particular ML endeavor or project where members of a single domain require near real-time access to the same underlying file storage and IDE. The user can access, read, edit, and share their notebooks in near real-time, which gives them the quickest path to start iterating with their peers.

To create a shared space, you must first designate a space default execution role which will govern the permissions for any user that utilizes the space. At the time of this writing, all users within a domain will have access to all shared spaces in their domain. Refer to Create a shared space for the latest documentation on adding shared spaces to an existing domain.

Set up your domain for IAM federation

Before setting up AWS Identity and Access Management (IAM) federation for your SageMaker Studio domain, you need to set up an IAM federation user role (such as a platform administrator) in your IdP, as discussed in the *Identity management* section.

For detailed instructions for setting up SageMaker Studio with the IAM option, refer to Onboard to Amazon SageMaker Domain Using IAM Identity Center.

Set up your domain for single sign-on (SSO) federation

To use single sign-on (SSO) federation, you need to enable AWS IAM Identity Center in your AWS Organizations management account in the same Region where you need to run SageMaker Studio. The domain setup steps are similar to IAM federation steps, except you select **AWS IAM Identity Center (IdC)** in the **Authentication** section.

For detailed instructions, refer to Onboard to Amazon SageMaker Domain Using IAM Identity Center.

SageMaker Studio user profile

A *user profile* represents a single user within a domain, and is the main way to reference a "person" for the purposes of sharing, reporting, and other user-oriented features. This entity is created when a user onboards toSageMaker Studio. If an administrator invites a person by email or imports them from IdC, a user profile is automatically created. A user profile is the primary holder of settings for an individual user, and has a reference to the user's private Amazon Elastic File System (Amazon EFS) home directory. We recommend creating a user profile for each physical user of the SageMaker Studio application. Each user has their own dedicated directory on Amazon EFS, and user profiles cannot be shared across domains in the same account.

Each user profile sharing the SageMaker Studio domain gets dedicated compute resource(s) (such as SageMaker Amazon Elastic Compute Cloud (Amazon EC2) instance(s)) to run notebooks. The compute instances allocated to user one are completely isolated from those allocated to user two. Similarly, the compute resources allocated to users in one AWS account are completely separate from those allocated to users in another account. Each user can run up to four applications (*apps*) within isolated Docker containers, or images on the same instance type.

Jupyter Server app

When you launch an Amazon SageMaker Studio notebook for a user by accessing the pre-signed URL or by logging in using AWS IAM IdC, the Jupyter Server app is launched in the SageMaker service-managed VPC instance. Each user gets their own dedicated Jupyter Server app in a private app. By default, the Jupyter Server app for SageMaker Studio notebooks is run on a dedicated `ml.t3.medium` instance (reserved as a *system* instance type). The compute for this instance is not billed to the customer.

The Jupyter Kernel Gateway app

The Kernel Gateway app can be created through the API or the SageMaker Studio interface, and it runs on the chosen instance type. This app can be run using one of the built-in SageMaker Studio images that are preconfigured with popular data science, and deep learning packages such as TensorFlow, Apache MXNet, and PyTorch.

Users can start and run multiple Jupyter notebook kernels, terminal sessions, and interactive consoles within the same SageMaker Studio image/Kernel Gateway app. Users can also run up to four Kernel Gateway apps or images on the same physical instance—each isolated by its container/image.

To create additional apps, you need to use a different instance type. A user profile can have only one instance running, of any instance type. For example, a user can run both a simple notebook using the SageMaker Studio built-in data science image, and another notebook using the built-in

TensorFlow image, on the same instance. Users are billed for the time the instance is running. To avoid costs when the user is not actively running SageMaker Studio, the user needs to shut down the instance. For more information, refer to Shut down and update Studio Apps.

Every time you shut down and reopen a Kernel Gateway app from the SageMaker Studio interface, that app is started on a new instance. This means that the package's installation is not persisted through restarts of the same app. Similarly, if a user changes the instance type on a notebook, their installed packages and session variables are lost. However, you can use features such as *bring your own image* and *lifecycle scripts* to bring the user's own packages to SageMaker Studio and persist them through instance switches and new instance launches.

Amazon Elastic File System volume

When a domain is created, a single Amazon Elastic File System (Amazon EFS) volume is created for use by all the users within the domain. Each user profile receives a private home directory within the Amazon EFS volume for storing the user's notebooks, GitHub repositories, and data files. Each space within a domain receives a private directory within the Amazon EFS volume that can be accessed by multiple user profiles. Access to the folders is segregated by user, through filesystem permissions. SageMaker Studio creates a global unique user ID for each user profile or space, and applies it as a Portable Operating System Interface (POSIX) user/group ID for the user's home directory on EFS, which prevents other users/spaces from accessing its data.

Backup and recovery

An existing EFS volume cannot be attached to a new SageMaker domain. In a production setting, make sure the Amazon EFS volume is backed up (to another EFS volume, or to Amazon Simple Storage Service (Amazon S3)). If an EFS volume is accidentally deleted, the administrator has to tear down and recreate the SageMaker Studio domain. The process is as follows:

Back up the list of user profiles, spaces and the associated EFS user IDs (UIDs) through the `ListUserProfiles`, `DescribeUserProfile`, `List Spaces`, and `DescribeSpace` API calls.

1. Create a new SageMaker Studio domain.
2. Create the user profiles and spaces.
3. For each user profile, copy over the files from the backup on EFS/Amazon S3.
4. Optionally, delete all apps and user profiles, on the old SageMaker Studio domain.

For detailed instructions refer to appendix section *SageMaker Studio domain backup and recovery*.

Note

This can also be achieved through `LifecycleConfigurations` to back up data to and from S3 every time a user starts their app.

Amazon EBS volume

An Amazon Elastic Block Store (Amazon EBS) storage volume is also attached to each SageMaker Studio Notebook instance. It's used as the root volume of the container or image running on the instance. While Amazon EFS storage is persistent, the Amazon EBS volume attached to container is temporary. The data stored locally on Amazon EBS volume won't be persisted if customer deletes the app.

Securing access to the pre-signed URL

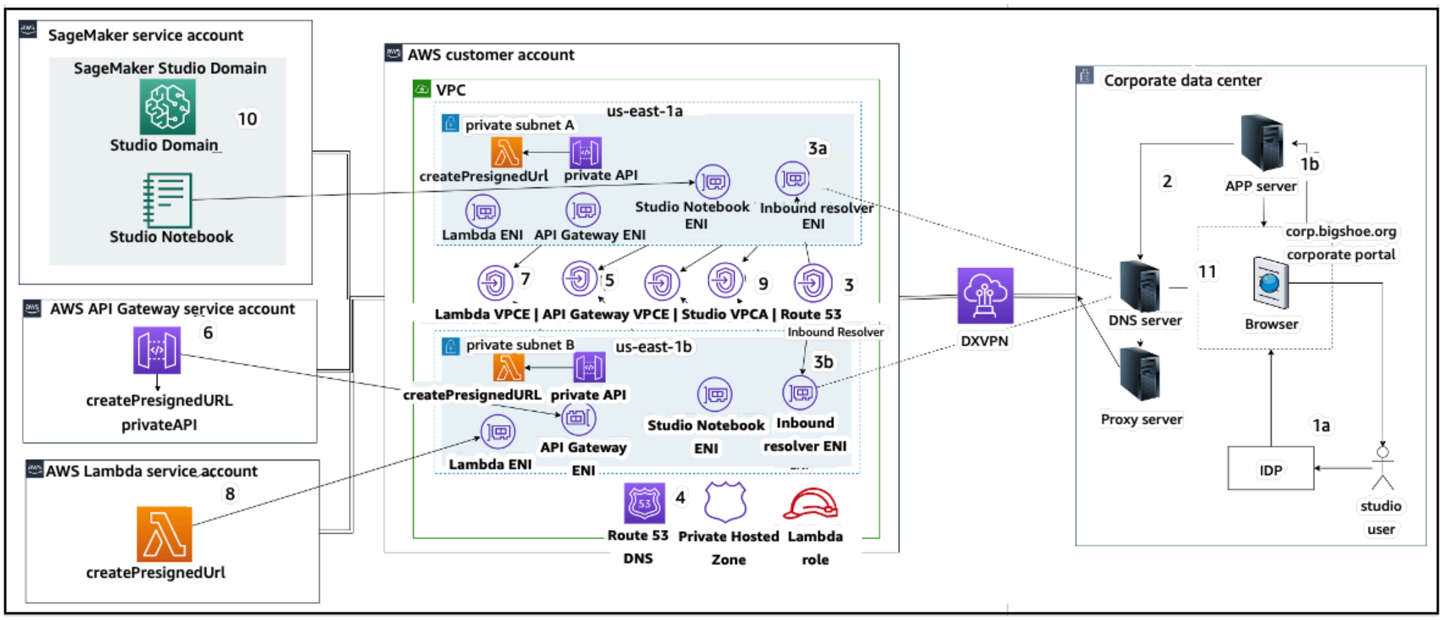
When a SageMaker Studio user opens the notebook link, SageMaker Studio validates the federated user's IAM policy to authorize access, and generates and resolves the pre-signed URL for the user. Because the SageMaker console runs on an internet domain, this generated, pre-signed URL is visible in the browser session. This presents an undesired threat vector for data theft and gaining access to customer data when proper access controls are not enforced.

Studio supports a few methods for enforcing access controls against pre-signed URL data theft:

- Client IP validation using the IAM policy condition `aws:sourceIp`
- Client VPC validation using the IAM condition `aws:sourceVpc`
- Client VPC endpoint validation using the IAM policy condition `aws:sourceVpce`

When you access SageMaker Studio notebooks from the SageMaker console, the only available option is to use client IP validation with the IAM policy condition `aws:sourceIp`. However, you can use browser traffic routing products such as Zscaler to ensure scale and compliance for your workforce internet access. These traffic routing products generate their own source IP, whose IP range is not controlled by the enterprise customer. This makes it impossible for these enterprise customers to use the `aws:sourceIp` condition.

To use client VPC endpoint validation using the IAM policy condition `aws:sourceVpce`, the creation of a pre-signed URL needs to originate in the same customer VPC where SageMaker Studio is deployed, and resolution of the pre-signed URL needs to happen via a SageMaker Studio VPC endpoint on the customer VPC. This resolution of the pre-signed URL during access time for corporate network users can be accomplished using DNS forwarding rules (both in Zscaler and corporate DNS), and then into the customer VPC endpoint using an [Amazon Route 53](https://aws.amazon.com/route53/) inbound resolver as shown in the following architecture:



Accessing Studio pre-signed URL with VPC endpoint over corporate network

For step-by-step guidance setting up the preceding architecture, refer to [Secure Amazon SageMaker Studio presigned URLs Part 1: Foundational infrastructure](https://aws.amazon.com/blogs/machine-learning/secure-amazon-sagemaker-studio-presigned-urls-part-1-foundational-infrastructure/).

SageMaker domain quotas and limits

- SageMaker Studio domain SSO federation is supported in only the Region, across member accounts of the AWS organization where AWS Identity Center is provisioned.
- Shared spaces are not currently supported with domains set up with AWS Identity Center.
- VPC and subnet configuration cannot be changed after creating the domain. You can, however, create a new domain with a different VPC and subnet configuration.
- Domain access cannot be switched between IAM and SSO modes after creating the domain. You can create a new domain with a different authentication mode.
- There is a limit of four kernel gateway apps per instance type launched for every user.

- Each user can launch only one instance of each instance type.
- There are limits on the resources consumed within a domain, such as number of instances launched by instance types, and number of user profiles that can be created. Refer to the service quota page for a complete list of service limits.
- Customers can submit an enterprise support case with business justification to raise the default resource limits such as number of domains or user profiles, subjected to account-level guardrails.
- The hard limit on the number of concurrent apps per account is 2,500 apps. Domains and user profile limits are dependent on this hard limit. For example, an account can have a single domain with 1,000 user profiles, or 20 domains with 50 user profiles each.

Identity management

This section discusses how workforce users in a corporate directory federate into AWS accounts and access SageMaker Studio. First, we will briefly describe how users, groups, and roles are mapped, and how user federation works.

Users, groups, and role

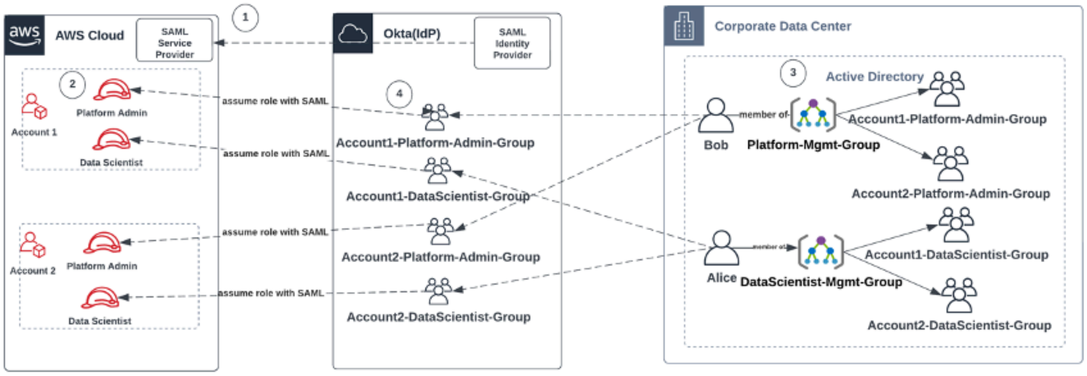
In AWS, resource permissions are managed using users, groups, and roles. Customers can manage their users and groups either through IAM, or in a corporate directory such as Active Directory (AD), enabled through an external IdP such as Okta, that allows them to authenticate the users to various applications running in the cloud and on-premises.

As discussed in the AWS Security Pillar Identity Management section, it is a best practice to manage your user identities in a central IdP, because this helps to easily integrate with your back-end HR processes, and helps to manage access to your workforce users.

IdPs such as Okta allow end users to authenticate to one or more AWS accounts and gain access to specific roles using SSO with security assertation markup language (SAML). IdP admins have the ability to download roles from AWS accounts into IdP, and assign those to users. When logging in to AWS, end users are presented with an AWS screen that displays a list AWS roles assigned to them in one or more AWS accounts. They can select the role to assume for login, which defines their permissions for the duration of that authenticated session.

A group must exist in IdP for each specific account and role combination that you want to provide access to. You can think of these groups as *AWS role-specific groups*. Any user who is a member of these role specific groups is granted a single entitlement: access to one specific role in one specific AWS account. However, this single entitlement process does not scale to manage user access by assigning each user to specific AWS role groups. To simplify administration, we recommend you also create a number of groups for all of the distinct user-sets in your organization that require different sets of AWS entitlements.

To illustrate the central IdP setup, consider an enterprise with AD setup, where users and groups are synchronized to the IdP directory. In AWS, these AD groups are mapped to IAM roles. The major steps of the workflow follow:



Workflow for onboarding AD users, AD groups and IAM roles

1. In AWS, Setup SAML integration for each of your AWS accounts with your IdP.
2. In AWS, set up roles in each AWS account and sync to IdP.
3. In the corporate AD system:
   a. Create an AD Group for each account role and sync to IdP (for example, `Account1-Platform-Admin-Group` (aka AWS Role Group)).
   b. Create a management group at each persona level (for example, `Platform-Mgmt-Group`) and assign AWS role groups as members.
   c. Assign users to that management group to allow access to AWS account roles.
4. In IdP, map AWS role groups (such as `Account1-Platform-Admin-Group`) to AWS account roles (such as Platform Admin in Account1).
5. When Data Scientist Alice logs in to Idp, they are presented with an AWS Federation App UI with two options to choose from: ‘Account 1 Data Scientist’ and ‘Account 2 Data Scientist’.
6. Alice chooses the ‘Account 1 Data Scientist’ option, and they are connected to their authorized application in AWS Account 1 (SageMaker Console).

For detailed instructions on setting up SAML account federation refer Okta’s How to Configure SAML 2.0 for AWS Account Federation.

User federation

Authentication for SageMaker Studio can either be done using IAM or IAM IdC. If the users are managed through IAM, they can choose the IAM mode. If the enterprise uses an external IdP, they can either federate through IAM or IAM IdC. Note that the authentication mode cannot be updated for an existing SageMaker Studio domain, so it is critical to make the decision before creating a production SageMaker Studio domain.

If SageMaker Studio is set up in IAM mode, SageMaker Studio users access the app through a pre-signed URL that automatically signs a user in to the SageMaker Studio app when accessed through a browser.

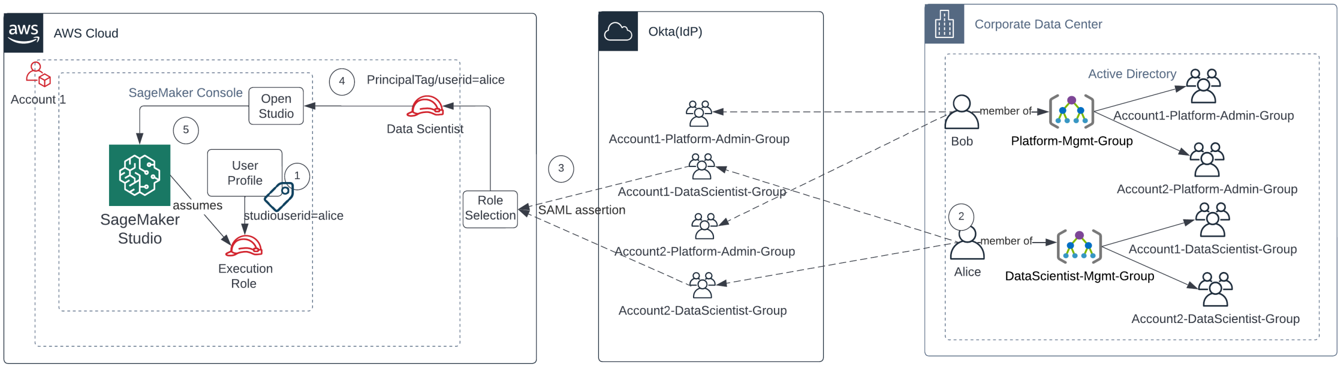
IAM users

For IAM users, the administrator creates SageMaker Studio user profiles for each user, and associates the user profile with an IAM role that allows the necessary actions that the user needs to perform from within Studio. To restrict an AWS user from accessing only their SageMaker Studio user profile, the administrator should tag the SageMaker Studio user profile and attach an IAM policy to the user that allows them to access only if the tag value is the same as the AWS user name. The policy statement looks like this:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AmazonSageMakerPresignedUrlPolicy",
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreatePresignedDomainUrl"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "sagemaker:ResourceTag/studiouserid": "${aws:username}"
                }
            }
        }
    ]
}
```

AWS IAM or account federation

The AWS account federation method enables customers to federate into the SageMaker Console from their SAML IdP, such as Okta. To restrict users from accessing only their user profile, the administrator should tag the SageMaker Studio user profile, add `PrincipalTags` on the IdP, and set them as transitive tags. The following diagram depicts how the federated user (Data Scientist Alice) is authorized to access their own SageMaker Studio user profile.



Accessing SageMaker Studio in IAM federtation mode

1. The Alice SageMaker Studio user profile is tagged with their user ID, and associated to execution role.
2. Alice authenticates to IdP (Okta).
3. IdP authenticates Alice and posts a SAML assertion with the two roles (Data Scientist for accounts 1 and 2) Alice is member of. Alice selects the Data Scientist role for account 1.
4. Alice is logged in to Account 1 SageMaker Console, with the assumed role of Data Scientist. Alice opens their Studio app instance from the list of studio app instances.
5. The Alice principal tag in the assumed role session is validated against the selected SageMaker Studio app instance user profile tag. If the profile tag is valid, the SageMaker Studio app instance is launched, assuming the execution role.

If you want to automate the creation of SageMaker Execution roles and policies as part of user onboarding, the following is one way to accomplish this:

1. Set up an AD group such as `SageMaker-Account1-Group` at each account and Studio Domain level.
2. Add SageMaker-Account1-Group to the user's group membership when you need to onboard a user to SageMaker Studio.

Set up an automation process that listens to the `SageMaker-Account1-Group` membership event, and use AWS APIs to create the role, policies, tags, and SageMaker Studio user profile based on their AD group memberships. Attach the role to the user profile. For a sample policy, refer to Prevent SageMaker Studio users from accessing other user profiles.

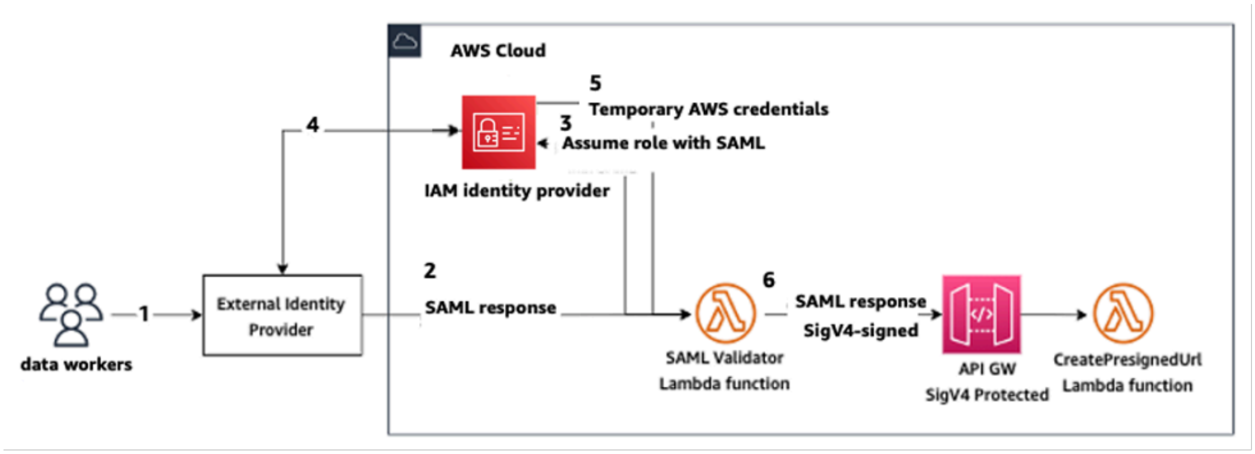
SAML authentication using AWS Lambda

In IAM mode, users can also be authenticated into SageMaker Studio using SAML assertions. In this architecture, the customer has an existing IdP, where they can create a SAML application for the users to access Studio (instead of AWS Identity Federation application). The customer's IdP is added to IAM. An AWS Lambda function helps validate the SAML assertion using IAM and STS, and then invokes an API gateway or a Lambda function directly, to create the pre-signed domain URL.

The advantage of this solution is that the Lambda function can customize logic for access to SageMaker Studio. For example:

- Automatically create a user profile if one does not exist.
- Attach or remove roles or policy documents to the SageMaker Studio execution role by parsing the SAML attributes.
- Customize the user profile by adding Life Cycle Configuration (LCC) and adding tags.

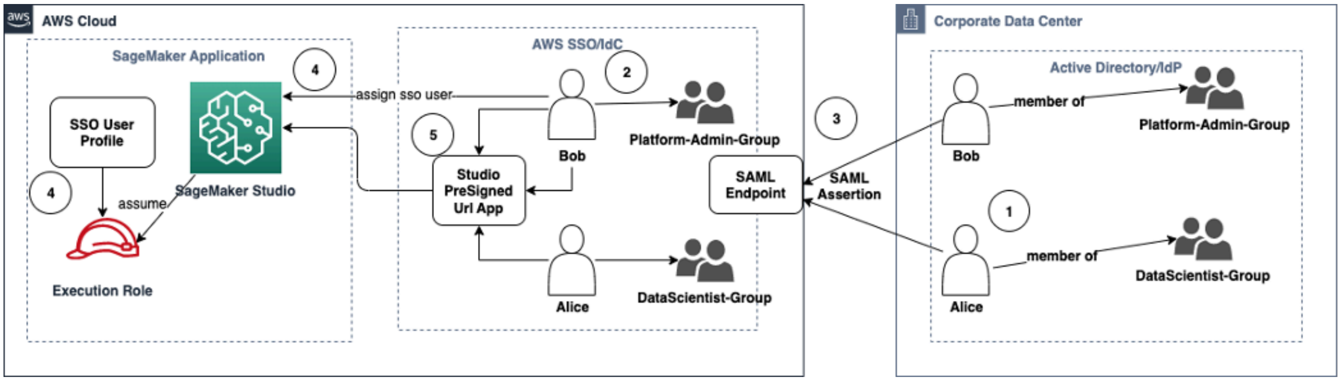
In summary, this solution will expose SageMaker Studio as a SAML2.0 application with custom logic for authentication and authorization. Refer the appendix section *SageMaker Studio access using SAML assertion* for implementation details.



Accessing SageMaker Studio using a custom SAML application

AWS IAM IdC federation

IdC federation method enables customers to federate directly into SageMaker Studio application from their SAML IdP (such as Okta). The following diagram depicts how the federated user is authorized to access their own SageMaker Studio instance.



Accessing SageMaker Studio in IAM IdC mode

1. In the corporate AD, the user is a member of AD groups such as the Platform Admin group and the Data Scientist group.
2. The AD user and AD groups from Identity Provider (IdP) are synced to AWS IAM Identity Center and available as single sign-on users and groups for assignments respectively.
3. The IdP posts a SAML assertion to the AWS IdC SAML endpoint.
4. In the SageMaker Studio, the IdC user is assigned to the SageMaker Studio application. This assignment can be done using IdC Group and SageMaker Studio will apply at each IdC user level. When this assignment is created, SageMaker Studio creates IdC user profile and attaches the domain execution role.
5. The user accesses the SageMaker Studio Application using the secure presigned URL hosted as a cloud application from the IdC. SageMaker Studio assumes the execution role attached to their IdC user profile.

Domain authentication guidance

Here are a few considerations when choosing a domain's authentication mode:

1. If you want your users to not access the AWS Management Console and view the SageMaker Studio UI directly, use single sign-on mode with AWS IAM IdC.

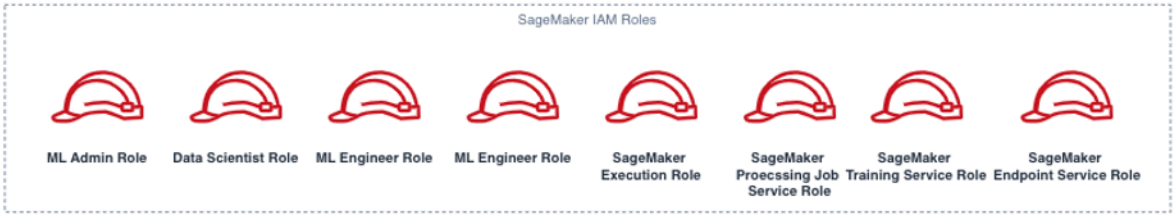
2. If you want your users to not access the AWS Management Console and view the SageMaker Studio UI directly in IAM mode, you can do that by using a Lambda function in the backend to generate a presigned URL for the user profile and redirecting them to the SageMaker Studio UI.
3. In IdC mode, each user is mapped to a single user profile.
4. All user profiles are automatically assigned the default execution role in IdC mode. If you would like your users to be assigned different execution roles, you will need to update the user profiles using the UpdateUserProfile API.
5. If you would like to restrict SageMaker Studio UI access in IAM mode (using the generated presigned URL) to a VPC endpoint, without traversing the internet, you can use a custom DNS resolver. Refer to the Secure Amazon SageMaker Studio presigned URLs Part 1: Foundational infrastructure blog post.

Permissions management

This section discusses the best practices for setting up commonly used IAM roles, policies, and guardrails for provisioning and operating the SageMaker Studio domain.

IAM roles and policies

As a best practice, you may want to first identify the relevant people and applications, known as principals involved in the ML lifecycle, and what AWS permissions you need to grant them. As SageMaker is a managed service, you also need to consider service principals which are AWS services that can make API calls on a user's behalf. The following diagram illustrates the different IAM roles you may want to create, corresponding to the different personas in the organization.



SageMaker IAM roles

These roles are described in detail, along with some examples of specific IAMpermissions they will need.

- **ML Admin user role** — This is a principal who provisions the environment for data scientists by creating studio domains and user profiles (`sagemaker:CreateDomain`, `sagemaker:CreateUserProfile`), creating AWS Key Management Service (AWS KMS) keys for users, creating S3 buckets for data scientists, and creating Amazon ECR repositories to house containers. They can also set default configurations and lifecycle scripts for users, build and attach custom images to the SageMaker Studio domain, and provide Service Catalog products such as custom projects, Amazon EMR templates.

  Because this principal will not run training jobs, for example, they don't need permissions to launch SageMaker training or processing jobs. If they're using infrastructure as code templates, such as CloudFormation or Terraform, to provision domains and users, this role would be assumed by the provisioning service to create the resources on the admin's behalf. This role may have read-only access to SageMaker using the AWS Management Console.

This user role will also need certain EC2 permissions to launch the domain inside a private VPC, KMS permissions to encrypt the EFS volume, as well as permissions to create a service linked role for Studio (`iam:CreateServiceLinkedRole`). We will describe those granular permissions later in the document.

- **Data Scientist user role —** This principal is the user logging in to SageMaker Studio, exploring the data, creating processing and training jobs and pipelines, and so on. The primary permission the user needs is permission to launch SageMaker Studio, and the rest of the policies can be managed by the SageMaker execution service role.
- **SageMaker execution service role —** Because SageMaker is a managed service, it launches jobs on a user's behalf. This role is often the broadest in terms of the allowed permissions, because many customers choose to use a single execution role to run training jobs, processing jobs, or model hosting jobs. While this is an easy way to get started, because customers mature in their journey, they often split the notebook execution role into separate roles for different API actions, especially when running those jobs in deployed environments.

  You associate a role with the SageMaker Studio domain upon creation. However, as customers may require the flexibility of having different roles associated with the different user profiles in the domain (for example, based on their job function), you can also associate a separate IAM role with each user profile. We recommend that you map a single physical user to a single user profile. If you don't attach a role to a user profile on creation, the default behavior is to associate the SageMakerStudio domain execution role with the user profile as well.

  In cases where multiple data scientists and ML engineers work together on a project and need a shared permission model for accessing resources, we recommend you create a team-level SageMaker service execution role for sharing the IAM permissions across your team members. In the instances where you need to lock down permissions at each user level, you can create an individual user-level SageMaker service execution role; however, you need to be mindful of your service limits.

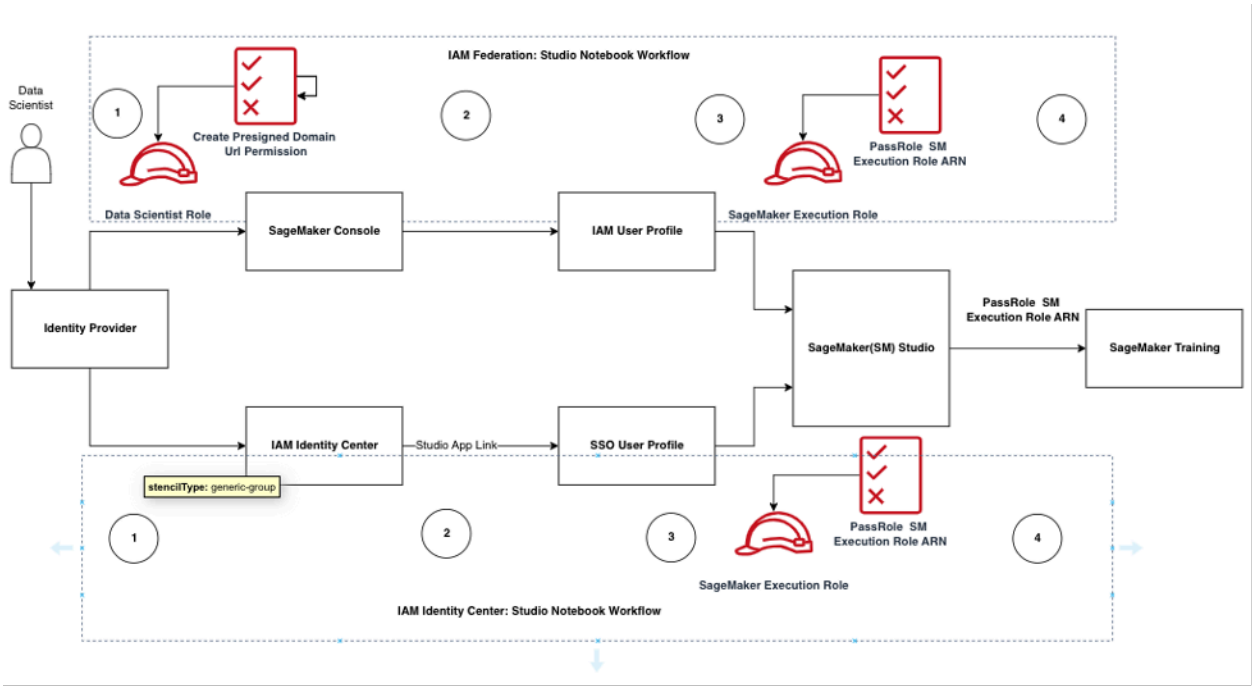
SageMaker Studio Notebook authorization workflow

This section, discusses how SageMaker Studio Notebook authorization works for various activities that the Data Scientist needs to perform for building and training the model right from the SageMaker Studio Notebook. The SageMaker domain supports two authorization modes:

- IAM federation

- IAM Identity Center

Next, this paper walks you through the Data Scientist authorization workflow for each of those modes.


IAM Federation: Studio Notebook Workflow
Data Scientist
1
Create Presigned Domain Url Permission
2
3
4
PassRole SM Execution Role ARN
Data Scientist Role
SageMaker Console
IAM User Profile
SageMaker Execution Role
Identity Provider
SageMaker(SM) Studio
PassRole SM Execution Role ARN
SageMaker Training
IAM Identity Center
Studio App Link
SSO User Profile
stencilType: generic-group
1
2
3
PassRole SM Execution Role ARN
4
SageMaker Execution Role
IAM Identity Center: Studio Notebook Workflow

Authentication and authorization workflow for Studio users

IAM Federation: SageMaker Studio Notebook workflows

1. A Data Scientist authenticates into their corporate identity provider and assumes the Data Scientist user role (the user federation role) in the SageMaker console. This federation role has `iam:PassRole` API permission on the SageMaker execution role to pass the role Amazon Resource Name (ARN) to SageMaker Studio.
2. The Data Scientist selects the **Open Studio** link from their Studio IAM user profile that is associated with the SageMaker execution role
3. The SageMaker Studio IDE service is launched, assuming the user profile's SageMaker execution role permissions. This role has `iam:PassRole` API permission on the SageMaker execution role to pass the role ARN to the SageMaker training service.
4. When Data Scientist launches the training job in the remote compute node(s), the SageMaker execution role ARN is passed to the SageMaker training service. This creates a new role session with this ARN and runs the training job. If you need to scope down the permission further for

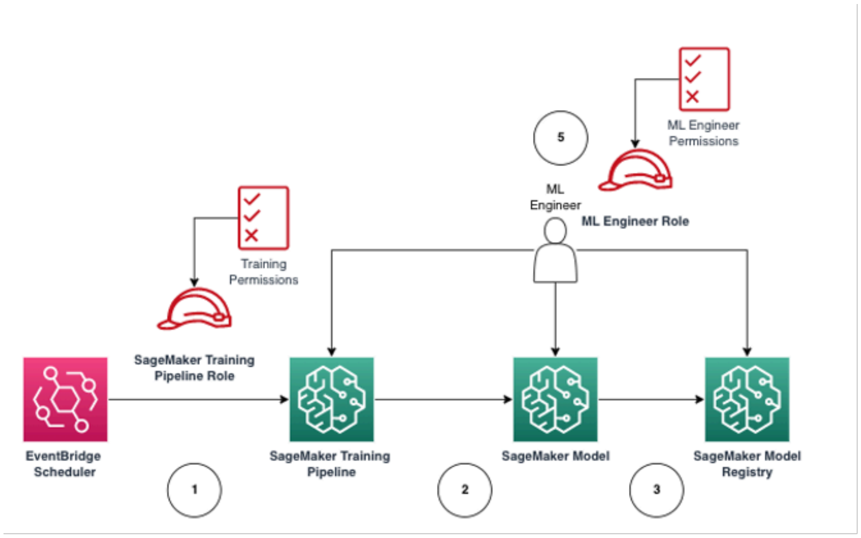
training job, you can create a training specific role and pass that role ARN when calling training API.

IAM Identity Center: SageMaker Studio Notebook workflow

1. The Data Scientist authenticates into their corporate identity provider and clicks on AWS IAM Identity Center. The Data Scientist is presented with Identity Center Portal for the user.
2. The Data Scientist clicks on the SageMaker Studio App link that was created from their IdC user profile, which is associated with the SageMaker execution role.
3. The SageMaker Studio IDE service is launched, assuming the user profile's SageMaker execution role permissions. This role has `iam:PassRole` API permission on the SageMaker execution role to pass the role ARN to the SageMaker training service.
4. When the Data Scientist launches the training job in remote compute node(s), the SageMaker execution role ARN is passed to the SageMaker training service. The execution role ARN creates new role session with this ARN, and runs the training job. If you need to scope down the permission further for training jobs, you can create a training-specific role and pass that role ARN when calling the training API.

Deployed environment: SageMaker training workflow

In deployed environments such as system testing and production, jobs are run through automated scheduler and event triggers, and human access to those environments are restricted from SageMaker Studio Notebooks. This section discusses how IAM roles work with the SageMaker training pipeline in the deployed environment.



SageMaker training workflow in a managed production environment

1. Amazon EventBridge scheduler triggers the SageMaker training pipeline job.
2. The SageMaker training pipeline job assumes the SageMaker training pipeline role to train the model.
3. The trained SageMaker model is registered into the SageMaker Model Registry.
4. An ML engineer assumes the ML engineer user role to manage the training pipeline and SageMaker model.

Data permissions

The ability for SageMaker Studio users to access any data source is governed by the permissions associated with their SageMaker IAM execution role. The policies attached can authorize them to read, write or delete from certain Amazon S3 buckets or prefixes, and connect to Amazon RDS databases.

Accessing AWS Lake Formation data

Many enterprises have begun using data lakes governed by AWS Lake Formation to enable fine grained data access for their users. As an example of such governed data, administrators can mask sensitive columns for some users while still enabling queries of the same underlying table.

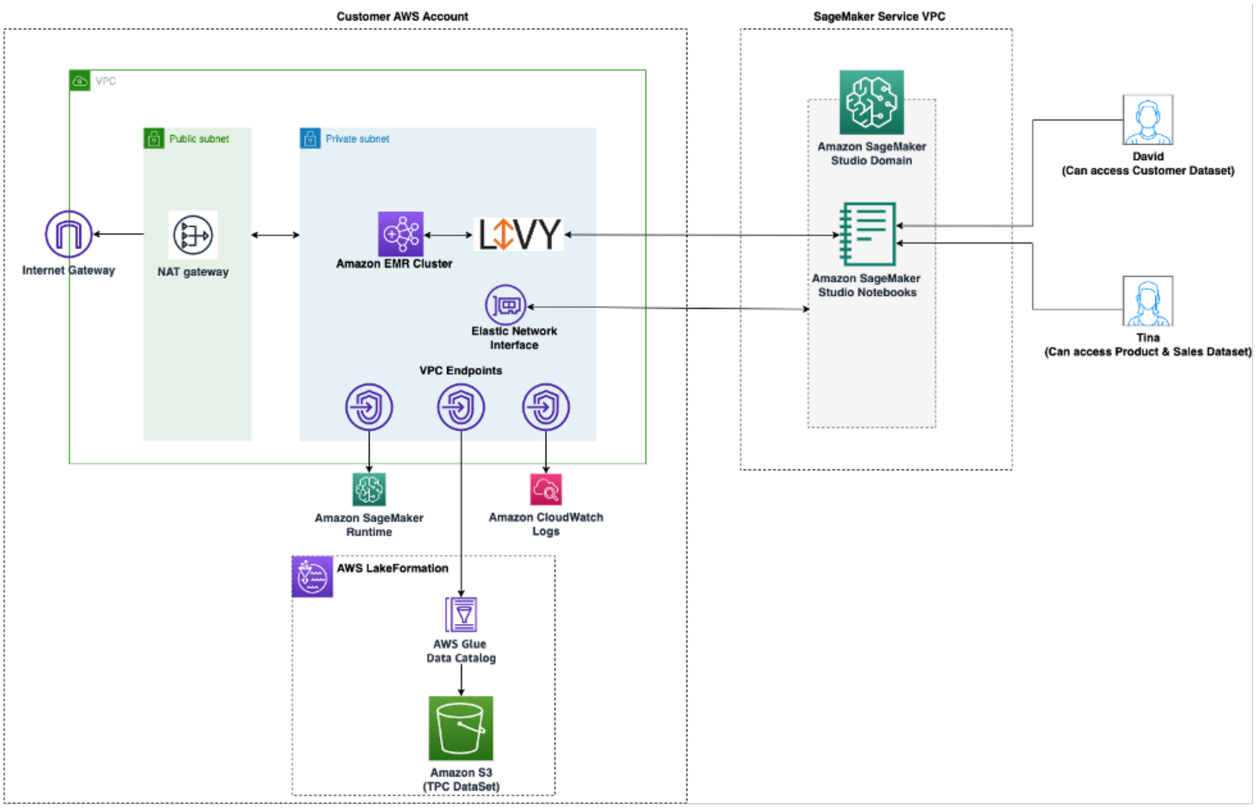
To utilize Lake Formation from SageMaker Studio, administrators can register SageMaker IAM execution roles as `DataLakePrincipals`. For more information, refer to Lake Formation

Permissions Reference. Once authorized, there are three primary methods for accessing and writing governed data from SageMaker Studio:

1. From a SageMaker Studio Notebook, users can utilize query engines such as Amazon Athena or libraries that build on top of boto3 to pull data directly to the notebook. The AWS SDK for Pandas (previously known as awswrangler) is a popular library. Following is a code example to show how seamless this can be:

```
transaction_id = wr.lakeformation.start_transaction(read_only=True)
df = wr.lakeformation.read_sql_query(
    sql=f"SELECT * FROM {table};",
    database=database,
    transaction_id=transaction_id
)
```

2. Use the SageMaker Studio native connectivity to Amazon EMR to read and write data at scale. Through use of Apache Livy and Amazon EMR runtime roles, SageMaker Studio has built native connectivity which allows you to pass your SageMaker execution IAM role (or other authorized role) to an Amazon EMR cluster for data access and processing. Refer to Connect to an Amazon EMR Cluster from Studio for up-to-date instructions.



Architecture for accessing data managed by Lake Formation from SageMaker Studio

3. Use the SageMaker Studio native connectivity to AWS Glue interactive sessions to read and write data at scale. SageMaker Studio Notebooks have built-in kernels that allow users to interactively run commands on AWS Glue. This enables the scalable use of Python, Spark, or Ray backends which can seamlessly read and write data at scale from governed data sources. The kernels allow users to pass their SageMaker execution or other authorized IAM roles. Refer to Prepare Data using AWS Glue Interactive Sessions for more information.

Common guardrails

This section discusses the most commonly-used guardrails for applying governance on your ML resources using IAM policies, resource policies, VPC endpoint policies, and service control policies (SCPs).

Limit notebook access to specific instances

This service control policy can be used to limit the instance types that data scientists have access to, while creating Studio notebooks. Note that any user will need the “system” instance allowed to create the default Jupyter Server app that hosts SageMaker Studio.

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Sid": "LimitInstanceTypesforNotebooks",
             "Effect": "Deny",
             "Action": [
                 "sagemaker:CreateApp"
             ],
             "Resource": "*",
             "Condition": {
                 "ForAnyValue:StringNotLike": {
                     "sagemaker:InstanceTypes": [
                         "ml.c5.large",
                         "ml.m5.large",
                         "ml.t3.medium",
                         "system"
                     ]
                 }
             }
         }
     ]
 }
```

Limit non-compliant SageMaker Studio domains

For SageMaker Studio domains, the following service control policy may be used to enforce traffic to access customer resources so they do not go over the public internet, but rather through a customer’s VPC:

```
{
     "Version": "2012-10-17",
     "Statement": [
         {   "Sid": "LockDownStudioDomain",
             "Effect": "Deny",
             "Action": [
                 "sagemaker:CreateDomain"
             ],
             "Resource": "*",
             "Condition": {
                          "StringNotEquals": {"sagemaker:AppNetworkAccessType":
 "VpcOnly"
                 },
                 "Null": {
                         "sagemaker:VpcSubnets": "true",
                         "sagemaker:VpcSecurityGroupIds": "true"
                 }
             }
         }
     ]
 }
```

Limit launching unauthorized SageMaker images

The following policy prevents a user from launching an unauthorized SageMaker image within their domain:f

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Action": [
                 "sagemaker:CreateApp"
              ],
             "Effect": "Allow",
             "Resource": "*",
             "Condition": {
                 "ForAllValues:StringNotLike": {
                     "sagemaker:ImageArns":
                         [
                         "arn:aws:sagemaker:*:*:image/{ImageName}"
                         ]
                 }
             }
         }
     ]
 }
```

Launch notebooks only via SageMaker VPC endpoints

In addition to VPC endpoints for the SageMaker control plane, SageMaker supports VPC endpoints for users to connect to SageMaker Studio notebooks or SageMaker notebook instances. If you have already set up a VPC endpoint for a SageMaker Studio/notebook instance, the following IAM condition key will only allow connections to SageMaker Studio notebooks if they are made via the SageMaker Studio VPC endpoint or via the SageMaker API endpoint.

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Sid": "EnableSageMakerStudioAccessviaVPCEndpoint",
             "Effect": "Allow",
             "Action": [
                 "sagemaker:CreatePresignedDomainUrl",
                 "sagemaker:DescribeUserProfile"
             ],
             "Resource": "*",
             "Condition": {
                 "ForAnyValue:StringEquals": {
                     "aws:sourceVpce": [
                         "vpce-111bbccc",
                         "vpce-111bbddd"
                     ]
                 }
             }
         }
     ]
 }
```

Limit SageMaker Studio notebook access to a limited IP range

Corporations will often limit SageMaker Studio access to certain allowed corporate IP ranges. The following IAM policy with the `SourceIP` condition key can limit this.

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Sid": "EnableSageMakerStudioAccess",
             "Effect": "Allow",
             "Action": [
                 "sagemaker:CreatePresignedDomainUrl",
                 "sagemaker:DescribeUserProfile"
             ],
             "Resource": "*",
             "Condition": {
                 "IpAddress": {
                     "aws:SourceIp": [
                         "192.0.2.0/24",
                         "203.0.113.0/24"
                     ]
                 }
             }
         }
     ]
 }
```

Prevent SageMaker Studio users from accessing other user profiles

As an administrator, when you create the user profile, ensure the profile is tagged with the SageMaker Studio user name with the tag key `studiouserid`. The principal (user or role attached to the user) should also have a tag with the key `studiouserid` (this tag can be named anything, and is not restricted to `studiouserid`).

Next, attach the following policy to the role the user will assume when launching SageMaker Studio.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AmazonSageMakerPresignedUrlPolicy",
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreatePresignedDomainUrl"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "sagemaker:ResourceTag/studiouserid": "${aws:PrincipalTag/
studiouserid}"
                }
            }
        }
    ]
}
```

Enforce tagging

Data scientists need to use SageMaker Studio notebooks to explore data, and build and train models. Applying tags to notebooks helps with monitoring usage and controlling costs, as well as ensuring ownership and auditability.

For SageMaker Studio apps, ensure the user profile is tagged. Tags are automatically propagated to apps from the user profile. To enforce user profile creation with tags (supported through CLI and SDK), consider adding this policy to the admin role:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnforceUserProfileTags",
            "Effect": "Allow",
            "Action": "sagemaker:CreateUserProfile",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": [
                        "studiouserid"
                    ]
                }
            }
        }
    ]
}
```

For other resources, such as training jobs and processing jobs, you can make tags mandatory using the following policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnforceTagsForJobs",
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreateTrainingJob",
                "sagemaker:CreateProcessingJob",
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": [
                        "studiouserid"
                    ]
                }
            }
        }
    ]
}
```

Root access in SageMaker Studio

In SageMaker Studio, the notebook runs in a Docker container which, by default, does not have root access to the host instance. Similarly, other than the default run-as user, all other user ID ranges inside the container are re-mapped as non-privileged user-IDs on the host instance itself. As a result, the threat of privilege escalation is limited to the notebook container itself.

When creating custom images, you might want to provide your user with non-root permissions for stricter controls; for example, avoiding running undesirable processes as root, or installing publicly-available packages. In such cases, you can create the image to run as a non-root user within the Dockerfile. Whether you create the user as root or non-root, you need to ensure that the UID/GID of the user is identical to the UID/GID in the AppImageConfig for the custom app, which creates the configuration for SageMaker to run an app using the custom image. For example, if your Dockerfile is built for a non-root user such as the following:

```
ARG NB_UID="1000"
ARG NB_GID="100"
...
USER $NB_UID
```

The `AppImageConfig` file needs to mention the same UID and GID in its `KernelGatewayConfig`:

```
{
    "KernelGatewayImageConfig": {
        "FileSystemConfig": {
            "DefaultUid": 1000,
            "DefaultGid": 100
        }
    }
}
```

The acceptable UID/GID values for custom images are 0/0 and 1000/100 for Studio images. For examples of building custom images and the associated `AppImageConfig` settings, refer to this Github repository.

To avoid users tampering with this, do not grant the `CreateAppImageConfig`, `UpdateAppImageConfig`, or `DeleteAppImageConfig` permissions to SageMaker Studio notebook users.

Network management

To set up the SageMaker Studio domain, you need to specify the VPC network, subnets, and security groups. When specifying the VPC and subnets, ensure that you allocate IPs considering the usage volume and expected growth that is discussed in the following sections.

VPC network planning

Customer VPC subnets associated to the SageMaker Studio domain must be created with the appropriate Classless Inter-domain Routing (CIDR) range, depending on the following factors:

- Number of users.
- Number of apps per user.
- Number of unique instance types per user.
- Average number of training instances per user.
- Expected growth percentage.

SageMaker and participating AWS services inject elastic network interfaces (ENI) into the customer VPC subnet for the following use cases:

- Amazon EFS injects an ENI for an EFS mount target for the SageMaker domain (one IP per subnet/Availability Zone attached to the SageMaker domain).
- SageMaker Studio injects an ENI for every unique instance used by a user profile or a shared space. For example:
  - If a user profile runs a *default* Jupyter server app (one 'system' instance), a *Data Science* app and a *Base Python* app (both running on an `ml.t3.medium` instance), Studio injects two IP addresses.
  - If a user profile runs a *default* Jupyter server app (one 'system' instance), a *Tensorflow GPU* app (on an `ml.g4dn.xlarge` instance), and a data wrangler app (on an `ml.m5.4xlarge` instance), Studio injects three IP addresses.
- An ENI for each VPC endpoint across domain VPC subnets/Availability Zones is injected (four IPs for SageMaker VPC endpoints; ~six IPs for participating services VPC endpoints such as S3, ECR, and CloudWatch.)

- If SageMaker training and processing jobs are launched with the same VPC configuration, each job needs two IP addresses per instance.

Note

VPC settings for SageMaker Studio, such as subnets and VPC-only traffic, do not get automatically passed on to the training/processing jobs created from SageMaker Studio. The user needs to set up VPC settings and network isolation as necessary when calling the Create*Job APIs. Refer to Run Training and Inference Containers in Internet-Free Mode for more information.

Scenario: Data scientist runs experiments on two different instance types

In this scenario, assume a SageMaker domain is set up in VPC-only traffic mode. There are VPC endpoints set up, such as SageMaker API, SageMaker runtime, Amazon S3, and Amazon ECR.

A data scientist is running experiments on Studio notebooks, running on two different instance types (for example, `ml.t3.medium` and `ml.m5.large`), and launching two apps in each instance type.

Assume the data scientist is also simultaneously running a training job with the same VPC configuration on an `ml.m5.4xlarge` instance.

For this scenario, the SageMaker Studio service will inject ENIs as follows:

Table 1 — ENIs injected into customer VPC for an experimentation scenario

Entity	Target	ENI injected	Notes	Level
EFS mount target	VPC subnets	Three	Three AZs/subnets	Domain
VPC endpoints	VPC subnets	30	Three AZs/subnets with 10 VPCE each	Domain
Jupyter Server	VPC subnet	One	One IP per instance	User

Entity	Target	ENI injected	Notes	Level
KernelGateway app	VPC subnet	Two	One IP per instance type	User
Training	VPC subnet	Two	Two IPs per training instance Five IPs per training instance if EFA is used	User

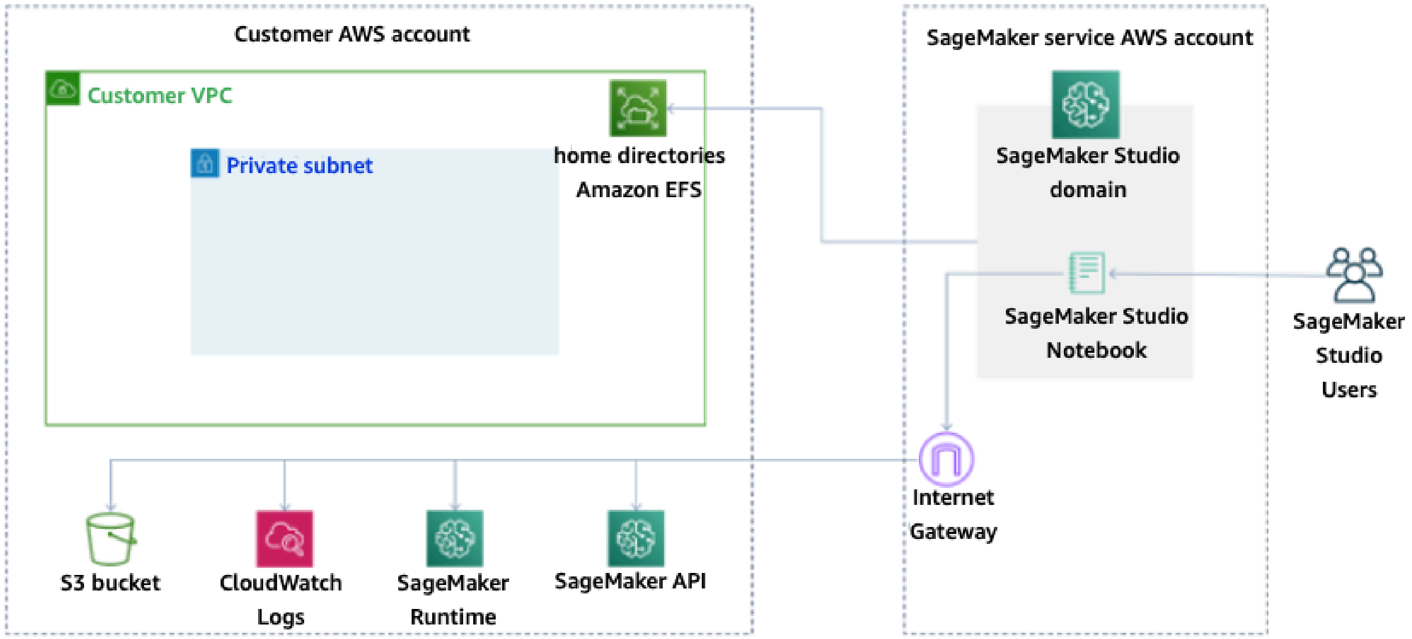
For this scenario, there are a total of 38 IPs consumed in the customer VPC where 33 IPs are shared across users at the domain level, and five IPs are consumed at the user level. If you have 100 users with similar user profiles in this domain performing these activities concurrently, then you will consume five x 100 = 500 IPs at the user level, on top of the domain level IP consumption, which is 11 IPs per subnet, for a total of 511 IPs. For this scenario, you need to create the VPC subnet CIDR with /22 that will allocate 1024 IP addresses, with room to grow.

VPC network options

A SageMaker Studio domain supports configuring the VPC network with one of the following options:

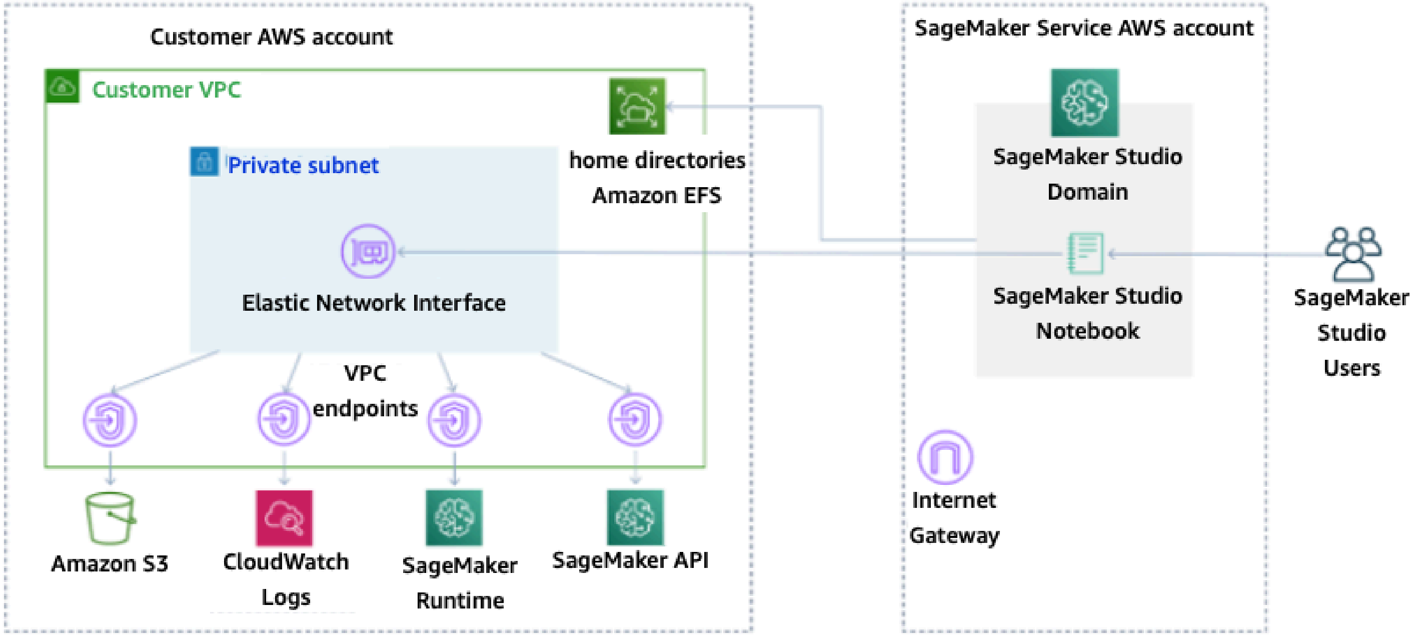
- Public internet only
- VPC only

The **public internet only** option allows SageMaker API services to use public internet via the internet gateway provisioned in the VPC, managed by the SageMaker service account, as seen in the following diagram:



Default mode: Internet access via SageMaker service account

The **VPC only** option disables internet routing from the VPC managed by the SageMaker service account, and allows customer to configure the traffic to be routed over VPC endpoints, as seen in the following diagram:



VPC only mode: No internet access via SageMaker service account

For a domain set up in VPC only mode, set up a security group per user profile to ensure complete isolation of underlying instances. Each domain in an AWS account can have its own VPC configuration and internet mode. For more details regarding setting up the VPC network configuration, refer to Connect SageMaker Studio Notebooks in a VPC to External Resources.

Limitations

- After a SageMaker Studio domain is created, you cannot associate new subnets to the domain.
- The VPC network type (*public internet only* or *VPC only)* cannot be changed.

Data protection

Before architecting an ML workload, the foundational practices that influence security should be in place. For example, data classification provides a way to categorize data based on levels of sensitivity, and encryption protects data by rendering it unintelligible to unauthorized access. These methods are important, because they support objectives such as preventing mishandling or complying with regulatory obligations.

SageMaker Studio provides several features for protecting data at rest and in-transit. However, as described in the AWS Shared Responsibility model, customers are responsible for maintaining control over the content that is hosted on AWS Global infrastructure. In this section, we describe how customers can use those features to protect their data.

Protect data at rest

To protect your SageMaker Studio notebooks along with your model-building data and model artifacts, SageMaker encrypts the notebooks, as well as the output from training and batch transform jobs. SageMaker encrypts these by default, using the AWS Managed Key for Amazon S3. This AWS Managed Key for Amazon S3 cannot be shared for cross-account access. For cross-account access, specify your customer-managed key while creating SageMaker resources so it can be shared for cross-account access.

With SageMaker Studio, data can be stored in the following locations:

- **S3 bucket** – When a shareable notebook is enabled, SageMaker Studio shares notebook snapshots and metadata in an S3 bucket.
- **EFS volume** – SageMaker Studio attaches an EFS volume to your domain for storing notebooks and data files. This EFS volume persists even after the domain is deleted.
- **EBS volume** – EBS is attached to the instance that the notebook runs on. This volume persists for the duration of the instance.

Encryption at rest with AWS KMS

- You can pass your AWS KMS key to encrypt an EBS volume attached to notebooks, training, tuning, batch transform jobs, and endpoints.

- If you don't specify a KMS key, SageMaker encrypts both operating system (OS) volumes and ML data volumes with a system-managed KMS key.
- Sensitive data that needs to be encrypted with a KMS key for compliance reasons should be stored in the ML storage volume or in Amazon S3, both of which can be encrypted using a KMS key you specify.

Protect data in transit

SageMaker Studio ensures that ML model artifacts and other system artifacts are encrypted in transit and at rest. Requests to the SageMaker API and console are made over a secure (SSL) connection. Some intra-network data in-transit (inside the service platform) is unencrypted. This includes:

- Command and control communications between the service control plane and training job instances (not customer data).
- Communications between nodes in distributed processing and training jobs (intra-network).

However, you can choose to encrypt communication between nodes in a training cluster. Enabling inter-container traffic encryption can increase training time, especially if you are using distributed deep learning algorithms.

By default, Amazon SageMaker runs training jobs in an Amazon VPC to help keep your data secure. You can add another level of security to protect your training containers and data by configuring a private VPC. Furthermore, you can configure your SageMaker Studio domain to run in VPC only mode, and set up VPC endpoints to route traffic over a private network without egressing traffic over the internet.

Data protection guardrails

Encrypt SageMaker hosting volumes at rest

Use the following policy to enforce encryption during hosting a SageMaker endpoint for online inference:

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
         "Sid": "Encryption",
         "Effect": "Allow",
         "Action": [
             "sagemaker:CreateEndpointConfig"
         ],
         "Resource": "*",
         "Condition": {
             "Null": {
                 "sagemaker:VolumeKmsKey": "false"
             }
         }
     }
   ]
 }
```

Encrypt S3 buckets used during Model Monitoring

Model Monitoring captures data sent to your SageMaker endpoint and stores it in an S3 bucket. When you set up the Data Capture Config, you need to encrypt the S3 bucket. Currently there is no compensating control for this.

In addition to capturing endpoint outputs, the Model Monitoring service checks for drift against a pre-specified baseline. You need to encrypt the outputs and the intermediate storage volumes used to monitor the drift.

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
         "Sid": "Encryption",
         "Effect": "Allow",
         "Action": [
             "sagemaker:CreateMonitoringSchedule",
             "sagemaker:UpdateMonitoringSchedule"
         ],
         "Resource": "*",
         "Condition": {
             "Null": {
                 "sagemaker:VolumeKmsKey": "false",
                 "sagemaker:OutputKmsKey": "false"
             }
         }
     }
   ]
 }
```

Encrypt a SageMaker Studio domain storage volume

Enforce encryption to storage volume attached to Studio domain. This policy requires a user to provide a CMK to encrypt the storage volumes attached to studio domains.

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Sid": "EncryptDomainStorage",
             "Effect": "Allow",
             "Action": [
                 "sagemaker:CreateDomain"
             ],
             "Resource": "*",
             "Condition": {
                 "Null": {
                     "sagemaker:VolumeKmsKey": "false"
                 }
             }
         }
     ]
 }
```

Encrypt data stored in S3 that is used to share notebooks

This is the policy to encrypt any data stored in the bucket that is used to share notebooks between users in a SageMaker Studio domain:

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Sid": "EncryptDomainSharingS3Bucket",
             "Effect": "Allow",
             "Action": [
                 "sagemaker:CreateDomain",
                 "sagemaker:UpdateDomain"
             ],
             "Resource": "*",
             "Condition": {
                 "Null": {
                     "sagemaker:DomainSharingOutputKmsKey": "false"
                 }
             }
         }
     ]
 }
```

Limitations

- Once a domain is created, you cannot update the attached EFS volume storage with a custom AWS KMS key.
- You cannot update training/processing jobs or endpoint configurations with KMS keys once they have been created.

Logging and monitoring

To help you debug your compilation jobs, processing jobs, training jobs, endpoints, transform jobs, notebook instances, and notebook instance lifecycle configurations, anything that an algorithm container, a model container, or a notebook instance lifecycle configuration sends to stdout or stderr is also sent to Amazon CloudWatch Logs. You can monitor SageMaker Studio using Amazon CloudWatch, which collects raw data and processes it into readable, near real-time metrics. These statistics are kept for 15 months, so you can access historical information and gain a better perspective on how your web application or service is performing.

Logging with CloudWatch

As the data science process is inherently experimental and iterative, it is essential to log activity such as notebook usage, training/processing job run time, training metrics, and endpoint serving metrics such as invocation latency. By default, SageMaker publishes metrics to CloudWatch Logs, and these logs can be encrypted with customer-managed keys using AWS KMS.

You can also use VPC endpoints to send logs to CloudWatch without using the public internet. You can also set alarms that watch for certain thresholds, and send notifications or take actions when those thresholds are met. For more information, refer to the Amazon CloudWatch User Guide.

SageMaker creates a single log group for Studio, under `/aws/sagemaker/studio`. Each user profile and app has their own log stream under this log group, and lifecycle configuration scripts have their own log stream as well. For example, a user profile named ‘studio-user’ with a Jupyter Server app and with an attached lifecycle script, and a Data Science Kernel Gateway app has the following log streams:

`/aws/sagemaker/studio/<domain-id>/studio-user/JupyterServer/default`

`/aws/sagemaker/studio/<domain-id>/studio-user/JupyterServer/default/LifecycleConfigOnStart`

`/aws/sagemaker/studio/<domain-id>/studio-user/KernelGateway/datascience-app`

For SageMaker to send logs to CloudWatch on your behalf, the caller of the Training/Processing/Transform job APIs will need the following permissions:

```
{
    "Version": "2012-10-17",
     "Statement": [
         {
             "Action": [
                 "logs:CreateLogDelivery",
                 "logs:CreateLogGroup",
                 "logs:CreateLogStream",
                 "logs:DeleteLogDelivery",
                 "logs:Describe*",
                 "logs:GetLogEvents",
                 "logs:GetLogDelivery",
                 "logs:ListLogDeliveries",
                 "logs:PutLogEvents",
                 "logs:PutResourcePolicy",
                 "logs:UpdateLogDelivery"
             ],
             "Resource": "*",
             "Effect": "Allow"
         }
     ]
 }
```

To encrypt those logs with a custom AWS KMS key, you will first need to modify the key policy to allow the CloudWatch service to encrypt and decrypt the key. Once you create a log encryption AWS KMS key, modify the key policy to include the following:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.region.amazonaws.com"
            },
            "Action": [
                "kms:Encrypt*",
                "kms:Decrypt*",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:Describe*"
            ],
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                    "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:region:account-
id:*"
                }
            }
        }
    ]
}
```

Note that you can always use `ArnEquals` and provide a specific Amazon Resource Name (ARN) for the CloudWatch log you want to encrypt. Here we are showing that you can use this key to encrypt all logs in an account for simplicity. Additionally, training, processing, and model endpoints publish metrics about the instance CPU and memory utilization, hosting invocation latency, and so on. You can further configure Amazon SNS to notify administrators of events when certain thresholds are crossed. The consumer of the training and processing APIs needs to have the following permissions:

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Action": [
                 "cloudwatch:DeleteAlarms",
                 "cloudwatch:DescribeAlarms",
                 "cloudwatch:GetMetricData",
                 "cloudwatch:GetMetricStatistics",
                 "cloudwatch:ListMetrics",
                 "cloudwatch:PutMetricAlarm",
                 "cloudwatch:PutMetricData",
                 "sns:ListTopics"
             ],
             "Resource": "*",
             "Effect": "Allow",
             "Condition": {
                 "StringLike": {
                     "cloudwatch:namespace": "aws/sagemaker/*"
                 }
             }

         },
         {
             "Action": [
                 "sns:Subscribe",
                 "sns:CreateTopic"
             ],
             "Resource": [
                 "arn:aws:sns:*:*:*SageMaker*",
                 "arn:aws:sns:*:*:*Sagemaker*",
                 "arn:aws:sns:*:*:*sagemaker*"
             ],
             "Effect": "Allow"
         }
     ]
 }
```

Audit with AWS CloudTrail

To improve your compliance posture, audit all your APIs with AWS CloudTrail. By default, all SageMaker APIs are logged with AWS CloudTrail. You do not need any additional IAM permissions to enable CloudTrail.

All SageMaker actions, with the exception of `InvokeEndpoint` and `InvokeEndpointAsync`, are logged by CloudTrail and are documented in the operations. For example, calls to the `CreateTrainingJob`, `CreateEndpoint`, and `CreateNotebookInstance` actions generate entries in the CloudTrail log files.

Every CloudTrail event entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS IAM user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service. For an example event, refer to the Log SageMaker API Calls with CloudTrail documentation.

By default, CloudTrail logs the Studio execution role name of the user profile as the identifier for each event. This works if each user has their own execution role. If multiple users share the same execution role, you can use the `sourceIdentity` configuration to propagate the Studio user profile name to CloudTrail. Refer to Monitoring user resource access from Amazon SageMaker Studio to enable the `sourceIdentity` feature. In a shared space, all actions refer to the space ARN as the source, and you cannot audit through `sourceIdentity`.

Cost attribution

SageMaker Studio has built in capabilities to help administrators track the spend of their individual domains, shared spaces, and users.

Automated tagging

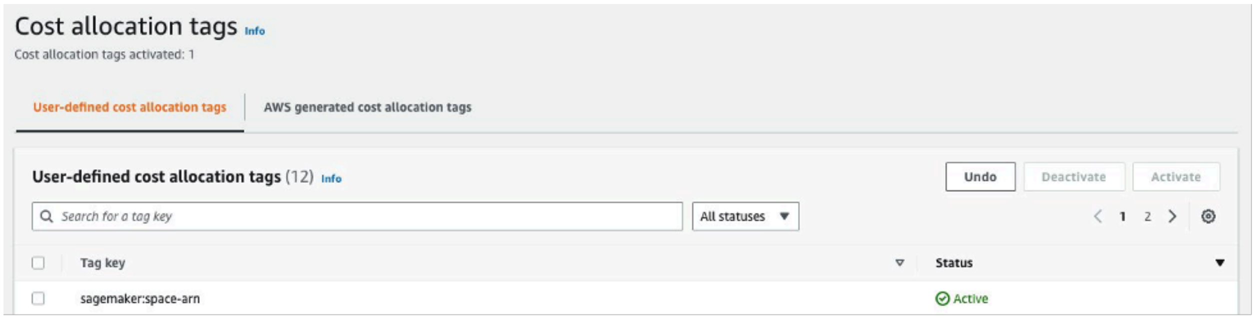
SageMaker Studio now automatically tags new SageMaker resources such as training jobs, processing jobs, and kernel apps with their respective `sagemaker:domain-arn`. At a more granular level, SageMaker also tags the resource with the `sagemaker:user-profile-arn` or `sagemaker:space-arn` to designate the the principal creator of the resource.

SageMaker domain EFS volumes are tagged with a key named `ManagedByAmazonSageMakerResource` with the value of the domain ARN. They do not have granular tags to understand the space usage on a per user level. Administrators can attach the EFS volume to an EC2 instance for bespoke monitoring though.

Cost monitoring

Automated tags enable Administrators to track, report, and monitor your ML spend through out-of-the-box solutions such as [AWS Cost Explorer]() and [AWS Budgets](), as well as custom solutions built on the data from [AWS Cost and Usage Reports]() (CURs).

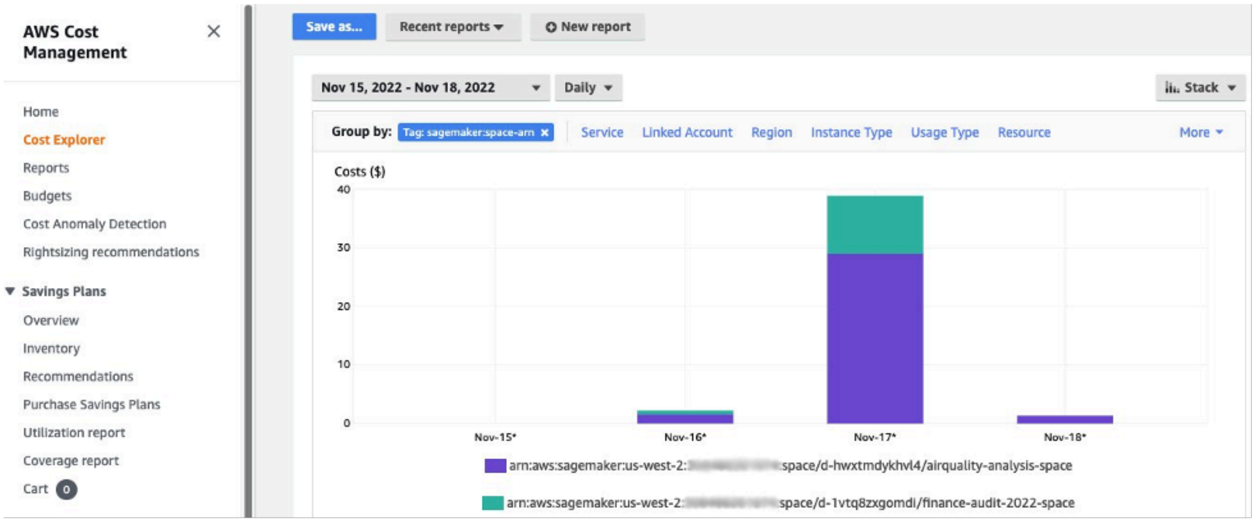
To use the attached tags for cost analysis, they must first be activated in the **[Cost allocation tags]()** section of the AWS Billing console. It can take up to 24 hours for tags to show up in the cost allocate tag panel, so you'll need to create a SageMaker resource prior to enabling them.


Cost allocation tags Info
Cost allocation tags activated: 1
User-defined cost allocation tags
AWS generated cost allocation tags
User-defined cost allocation tags (12) Info
Undo
Deactivate
Activate
Search for a tag key
All statuses
1
2

Tag key	Status
sagemaker:space-arn	Active

Space ARN enabled as cost allocation tags on Cost Explorer

After you have enabled a cost allocation tag, AWS will begin tracking your tagged resources, and after 24-48 hours, the tags will show up as selectable filters in cost explorer.



Costs grouped by shared space for a sample domain

Cost control

When the first SageMaker Studio user is onboarded, SageMaker creates an EFS volume for the domain. Storage costs are incurred for this EFS volume as notebooks and data files are stored in the user's home directory. When the user launches Studio notebooks, they are launched for the compute instances running the notebooks. Refer to Amazon SageMaker pricing for detailed breakdown of costs.

Administrators can control compute costs by specifying the list of instances a user can spin up, using IAM policies as mentioned in the *Common guardrails* section. In addition, we recommend that customers make use of the SageMaker Studio auto shutdown extension to save costs by automatically shutting down idle apps. This server extension periodically polls for running apps per user profile, and shuts down idle apps based on a timeout set by the administrator.

To set this extension for all users in your domain, you can use a lifecycle configuration as described in *Customization* section. Additionally, you can also use the extension checker to ensure all of your domain's users have the extension installed.

Customization

Lifecycle configuration

Lifecycle configurations are shell scripts initiated by SageMaker Studio lifecycle events, such as starting a new SageMaker Studio notebook. You can use these shell scripts to automate customization for your SageMaker Studio environments, such as installing custom packages, Jupyter extension for auto-shutdown of inactive notebook apps, and setting up Git configuration. For detailed instructions on how to build lifecycle configurations, refer to this blog: Customize Amazon SageMaker Studio using Lifecycle Configurations.

Custom images for SageMaker Studio notebooks

Studio notebooks come with a set of pre-built images, which consist of the Amazon SageMaker Python SDK and the latest version of the IPython runtime or kernel. With this feature, you can bring your own custom images to Amazon SageMaker notebooks. These images are then available to all users authenticated into the domain.

Developers and data scientists may require custom images for several different use cases:

- Access to specific or latest versions of popular ML frameworks such as TensorFlow, MXNet, PyTorch, or others.
- Bring custom code or algorithms developed locally to SageMaker Studio notebooks for rapid iteration and model training.
- Access to data lakes or on-premises data stores via APIs. Admins need to include the corresponding drivers within the image.
- Access to a backend runtime (also called kernel), other than IPython (such as R, Julia, or others). You can also use the approach outlined to install a custom kernel.

For detailed instructions on how to build a custom image, refer to Create a custom SageMaker image.

JupyterLab extensions

With SageMaker Studio JuypterLab 3 Notebook, you can take advantage of the ever-growing community of open-source JupyterLab extensions. This section highlights a few that fit naturally into the SageMaker developer workflow, but we encourage you to browse the available extensions or even create your own.

JupyterLab 3 now makes the process of packaging and installing extensions significantly easier. You can install the aforementioned extensions through bash scripts. For example, in SageMaker Studio, open the system terminal from the Studio launcher and run the following commands. In addition, you can automate the installation of these extensions using lifecycle configurations so they're persisted between Studio restarts. You can configure this for all the users in the domain or at an individual user level.

For example, to install an extension for an Amazon S3 file browser, run the following commands in the system terminal and be sure the refresh your browser:

```
conda init
conda activate studio
pip install jupyterlab_s3_browser
jupyter serverextension enable --py jupyterlab_s3_browser
conda deactivate
restart-jupyter-server
```

For more information on extension management, including how to write lifecycle configurations that work for both versions 1 and 3 of JupyterLab notebooks for backward compatibility, refer to Installing JupyterLab and Jupyter Server extensions.

Git repositories

SageMaker Studio comes pre-installed with a Jupyter Git extension for users to enter a bespoke URL of a Git repository, clone it to your EFS directory, push changes, and view commit history. Administrators can configure suggested git repos at the domain level so that they show up as drop-down selections for the end users. Refer to Attach Suggested Git Repos to Studio for up-to-date instructions.

If a repository is private, the extension will ask the user to enter their credentials into the terminal using the standard git installation. Alternatively, the user can store ssh credentials on their individual EFS directory for easier management.

Conda environment

SageMaker Studio notebooks use Amazon EFS as a persistent storage layer. Data scientists can make use of the persistent storage to create custom conda environments and use these environments to create kernels. These kernels are backed by EFS, and are persistent between kernel, app, or Studio restarts. Studio automatically picks up all valid environments as KernelGateway kernels.

The process to create a conda environment is straightforward for a data scientist, but the kernels take about a minute to populate on the kernel selector. To create an environment, run the following in a system terminal:

```
mkdir -p ~/.conda/envs
conda create --yes -p ~/.conda/envs/custom
conda activate ~/.conda/envs/custom
conda install -y ipykernel
conda config --add envs_dirs ~/.conda/envs
```

For detailed instructions, refer to the *Persist Conda environments to the Studio EFS volume* section in Four approaches to manage Python packages in Amazon SageMaker Studio notebooks.

Conclusion

In this whitepaper, we reviewed several best practices across areas such as operating model, domain management, identity management, permissions management, network management, logging, monitoring, and customization to enable platform administrators to set up and manage SageMaker Studio Platform.

Appendix

Multi-tenancy comparison

Table 2 — Multi-tenancy comparison

Multi-domain	Multi-account	Attribute-based access control (ABAC) within a single domain
Resource isolation is achieved using tags. SageMaker Studio automatically tags all resources with the domain ARN and user profile/ space ARN.	Each tenant is in their own account, so there is absolute resource isolation.	Resource isolation is achieved using tags. Users have to manage the tagging of created resources for ABAC.
List APIs cannot be restricte d by tags. UI filtering of resources is done on shared spaces, however, List API calls made through the AWS CLI or the Boto3 SDK will list resources across the Region.	List APIs isolation is also possible, since tenants are in their dedicated accounts.	List APIs cannot be restricte d by tags. List API calls made through the AWS CLI or the Boto3 SDK will list resources across the Region.
SageMaker Studio compute and storage costs per tenant can be easily monitored by using Domain ARN as a cost allocation tag.	SageMaker Studio compute and storage costs per tenant are easy to monitor with a dedicated account.	SageMaker Studio compute costs per tenant need to be calculated using custom tags. SageMaker Studio storage costs cannot be monitored per domain since all tenants share the same EFS volume.
Service quotas are set at the account level, so a single	Service quotas can be set at the account level for each tenant.	Service quotas are set at the account level, so a single

Multi-domain	Multi-account	Attribute-based access control (ABAC) within a single domain
tenant could still use up all resources.		tenant could still use up all resources.
Scaling to multiple tenants can be achieved through infrastructure as code (IaC) or Service Catalog.	Scaling to multiple tenants involve Organizations and vending multiple accounts.	Scaling needs a tenant specific role for each new tenant, and user profiles need to be manually tagged with tenant names.
Collaboration between users within a tenant is possible through shared spaces.	Collaboration between user within a tenant is possible through shared spaces.	All tenants will have access to the same shared space for collaboration.

SageMaker Studio domain backup and recovery

In the event of an accidental EFS delete, or when a domain needs to be recreated due to changes in networking or authentication, follow these instructions.

Option 1: Back up from existing EFS using EC2

SageMaker Studio domain backup

1. List user profiles and spaces in SageMaker Studio (CLI, SDK).
2. Map user profiles/spaces to UIDs on EFS.
    a. For each user in list of users/spaces, describe the user profile/space (CLI, SDK).
    b. Map user profile/space to `HomeEfsFileSystemUid`.
    c. Map user profile to `UserSettings['ExecutionRole']` if users have distinct execution roles.
    d. Identify the default Space execution role.
3. Create a new domain and specify the default Space execution role.
4. Create user profiles and spaces.
    - For each user in list of users, create user profile (CLI, SDK) using the execution role mapping.

5. Create a mapping for the new EFS and UIDs.
   a. For each user in list of users, describe user profile (CLI, SDK).
   b. Map user profile to `HomeEfsFileSystemUid`.
6. Optionally, delete all apps, user profiles, spaces, and then delete the domain.

EFS backup

To back up EFS, use the following instructions:

1. Launch the EC2 instance, and attach the old SageMaker Studio domain's inbound/outbound security groups to the new EC2 instance (allow NFS traffic over TCP on port 2049. Refer to Connect SageMaker Studio Notebooks in a VPC to External Resources.
2. Mount the SageMaker Studio EFS volume to the new EC2 instance. Refer to Mounting EFS file systems.
3. Copy over the files to EBS local storage: `>sudo cp -rp /efs /studio-backup:`
   a. Attach the new domain security groups to the EC2 instance.
   b. Mount the new EFS volume to the EC2 instance.
   c. Copy files to the new EFS volume.
   d. For each user in user's collection:
      i. Create the directory: `mkdir new_uid`.
      ii. Copy files from old UID directory to new UID directory.
      iii. Change ownership for all files: `chown <new_UID>` for all files.

Option 2: Back up from existing EFS using S3 and lifecycle configuration

1. Refer to Migrate your work to an Amazon SageMaker notebook instance with Amazon Linux 2.
2. Create an S3 bucket for backup (such as `>studio-backup`.
3. List all user profiles with execution roles.
4. In the current SageMaker Studio domain, set a default LCC script at the domain level.
   - In the LCC, copy everything in `/home/sagemaker-user` to the user profile prefix in S3 (for example, `s3://studio-backup/studio-user1`).
5. Restart all default Jupyter Server apps (for the LCC to be run).

6. Delete all apps, user profiles, and domains.
7. Create a new SageMaker Studio domain.
8. Create new user profiles from the list of user profiles and execution roles.
9. Set up an LCC at the domain level:
    - In the LCC, copy everything in the user profile prefix in S3 to `/home/sagemaker-user`
10. Create default Jupyter Server apps for all users with the LCC configuration (CLI, SDK).

SageMaker Studio access using SAML assertion

Solution setup:

1. Create a SAML application in your external IdP.
2. Set up the external IdP as an Identity Provider in IAM.
3. Create a `SAMLValidator` Lambda function that can be accessed by the IdP (through a function URL or API Gateway).
4. Create a `GeneratePresignedUrl` Lambda function and an API Gateway to access the function.
5. Create an IAM role that users can assume to invoke the API Gateway. This role should be passed in SAML assertion as an attribute in the following format:
    - Attribute name: https://aws.amazon.com/SAML/Attributes/Role
    - Attribute value: `<IdentityProviderARN>`, `<RoleARN>`
6. Update the SAML Assertion Consumer Service (ACS) endpoint to the `SAMLValidator` invoke URL.

SAML validator example code:

```
import requests
import os
import boto3
from urllib.parse import urlparse, parse_qs
import base64
import requests
from aws_requests_auth.aws_auth import AWSRequestsAuth
import json
```

```
# Config for calling AssumeRoleWithSAML
idp_arn = "arn:aws:iam::0123456789:saml-provider/MyIdentityProvider"
api_gw_role_arn = 'arn:aws:iam:: 0123456789:role/APIGWAccessRole'
studio_api_url = "abcdef.execute-api.us-east-1.amazonaws.com"
studio_api_gw_path = "https://" + studio_api_url + "/Prod "

# Every customer will need to get SAML Response from the POST call
def get_saml_response(event):
    saml_response_uri = base64.b64decode(event['body']).decode('ascii')
    request_body = parse_qs(saml_response_uri)
    print(f"b64 saml response: {request_body['SAMLResponse'][0]}")
    return request_body['SAMLResponse'][0]


def lambda_handler(event, context):
    sts = boto3.client('sts')

    # get temporary credentials
    response = sts.assume_role_with_saml(
                    RoleArn=api_gw_role_arn,
                    PrincipalArn=durga_idp_arn,
                    SAMLAssertion=get_saml_response(event)
                )
    auth = AWSRequestsAuth(aws_access_key=response['Credentials']['AccessKeyId'],
                      aws_secret_access_key=response['Credentials']['SecretAccessKey'],
                      aws_host=studio_api_url,
                      aws_region='us-west-2',
                      aws_service='execute-api',
                      aws_token=response['Credentials']['SessionToken'])

    presigned_response = requests.post(
        studio_api_gw_path,
        data=saml_response_data,
        auth=auth)

    return presigned_response
```

Further reading

- Setting up secure, well-governed machine learning environments on AWS (AWS blog)
- Configuring Amazon SageMaker Studio for teams and groups with complete resource isolation (AWS blog)
- Onboarding Amazon SageMaker Studio with AWS SSO and Okta Universal Directory (AWS blog)
- How to Configure SAML 2.0 for AWS Account Federation (Okta documentation)
- Build a Secure Enterprise Machine Learning Platform on AWS (AWS technical guide)
- Customize Amazon SageMaker Studio using Lifecycle Configurations (AWS blog)
- Bringing your own custom container image to Amazon SageMaker Studio notebooks (AWS blog)
- Build Custom SageMaker Project Templates – Best Practices (AWS blog)
- Multi-account model deployment with Amazon SageMaker Pipelines (AWS blog)
- Part 1: How NatWest Group built a scalable, secure, and sustainable MLOps platform (AWS blog)
- Secure Amazon SageMaker Studio presigned URLs Part 1: Foundational infrastructure (AWS blog)

Contributors

Contributors to this document include:

- Ram Vittal, ML Solutions Architect, Amazon Web Services
- Sean Morgan, ML Solutions Architect, Amazon Web Services
- Durga Sury, ML Solutions Architect, Amazon Web Services

Special thanks to the following who contributed ideas, revisions, and perspectives:

- Alessandro Cerè, AI/ML Solutions Architect, Amazon Web Services
- Sumit Thakur, SageMaker Product Leader, Amazon Web Services
- Han Zhang, Sr. Software Development Engineer, Amazon Web Services
- Bhadrinath Pani, Software Development Engineer, Amazon Web Services, Amazon Web Services

Document revisions

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Whitepaper updated	Broken links fixed and numerous editorial changes throughout.	April 25, 2023
Initial publication	Whitepaper published.	October 19, 2022

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.